NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN ARCHITECTURE FOR THE SEMANTIC PROCESSING OF NATURAL LANGUAGE INPUT TO A POLICY WORKBENCH

by

E. John Custy

March 2003

Thesis Advisor:	J. Bret Michael
Co-Advisor:	Neil C. Rowe

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: An Architecture for the Semantic Processing of Natural Language Input to a Policy Workbench			**5. FUNDING NUMBERS**
6. AUTHOR(S) E. John Custy			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

Formal methods hold significant potential for automating the development, refinement, and implementation of policy. For this potential to be realized, however, improved techniques are required for converting natural-language statements of policy into a computational form. In this paper we present and analyze an architecture for carrying out this conversion. The architecture employs semantic networks to represent both policy statements and objects in the domain of those statements. We present a case study which illustrates how a system based on this architecture could be developed. The case study consists of an analysis of natural language policy statements taken from a policy document for web sites at a university, and is carried out with support from a software tool we developed which converts text output from a natural language parser into a graphical form.

14. SUBJECT TERMS Policy, Natural Language Processing, Semantic Networks, Policy Workbench, Prolog			**15. NUMBER OF PAGES** 107
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for Public Release; distribution is unlimited

AN ARCHITECTURE FOR THE SEMANTIC PROCESSING OF NATURAL LANGUAGE INPUT TO A POLICY WORKBENCH

E. John Custy
B.S.E.E. New Jersey Institute of Technology, 1986
M.A. Cognitive and Neural Systems, Boston University, 1991
Master of Engineering, Engineering Science, The Pennsylvania State University, 1994

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2003

Author:

E. John Custy

Approved by:

James Bret Michael, Co-Advisor

Neil C. Rowe, Co-Advisor

Valdis Berzins, Chairman
Software Engineering Curriculum

Peter J. Denning, Chairman
Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Formal methods hold significant potential for automating the development, refinement, and implementation of policy. For this potential to be realized, however, improved techniques are required for converting natural-language statements of policy into a computational form. In this paper we present and analyze an architecture for carrying out this conversion. The architecture employs semantic networks to represent both policy statements and objects in the domain of those statements. We present a case study which illustrates how a system based on this architecture could be developed. The case study consists of an analysis of natural language policy statements taken from a policy document for web sites at a university, and is carried out with support from a software tool we developed which converts text output from a natural language parser into a graphical form.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Policy plays an important role in determining the success of business, professional, political and social organizations. During normal day-to-day operations, the behavior of any organization of nontrivial size reflects the decisions and actions performed under various circumstances by the individuals within that organization, and these decisions and actions are often influenced or even dictated by the organizations policy. Whether an implicit part of the work culture, or explicitly developed, communicated and enforced by management, policy plays an important part in determining the behavior, and thus ultimately the success, of the organization.

A given set of policy statements should imply logical consequences that, under ideal circumstances, would be easy to determine. Conceptually, it should be possible to determine whether a particular action under a particular circumstance conforms to or violates policy by simple examination of the policy statements themselves [Sibley, Michael, Wexelblat, 1991]. In practice, however, the full consequences of a policy set may be unclear because of large numbers of policy statements, the complexity that arises when those statements interact with each other, and the presence of implicit policy statements. Furthermore, it is not uncommon for policy sets to be internally inconsistent, to have "gaps" (that is, to leave important real-world circumstances unaccounted for), or to simply be unwieldy to communicate or work with ([Michael, Ong, Rowe, p. 1], [Stone et. al., p. 10]). The use of policy to manage networks and other complex distributed systems [Damianou, 2002] is of growing importance, and as these techniques mature we can expect that creating, refining, and communicating policy for them will become ever more challenging.

To deal with these problems, the concept of a policy workbench has been proposed [Sibley, Michael, Wexelblat, 1991]. As a collection of tools for supporting the development, refinement, communication and enforcement of policy, a policy workbench exploits the formal logical structure implicit in any collection of policies. Though the policy workbench concept has significant potential, this potential is not being realized, in part because of human-machine interface issues. In particular, the transformation of

policy statements from natural language into an equivalent representation appropriate for formal analysis is a labor-intensive and error-prone task requiring skills in both formal methods and the application area within which the policy statements apply. This difficulty is exacerbated by the fact that policies are constantly evolving, so that a policy database must be updated on a regular basis.

This paper investigates the use of natural language processing (NLP) techniques for converting general statements of policy from a plain-text natural-language form into an equivalent computational form that can be processed using formal methods. Building upon the work of previous researchers, we propose an architecture for representing and processing the semantic content of policies. The architecture is distinguished by the use of semantic networks as a representation language for both statements of policy and objects within the domain of policy. Conformance to, or violation of, the policy set is established by determining whether a semantic network representation of a policy statement is a sub-semantic network of the semantic network representing a given domain object. We present a case study in which statements of policy taken from a web-site policy database are mapped to a graphical form, and these representations are studied to identify potential problems and opportunities associated with the conversion of policy statements and domain objects into semantic networks.

The remainder of this paper is structured as follows. Section 2 introduces the topics of computer support for policy, the policy workbench, and natural-language processing. Section 3 describes related work carried out by other researchers and presents background information on natural-language processing. Section 4 presents our architecture and investigates design issues through a case study. Section 5 discusses the results of the case study, and Section 6 presents conclusions. Appendix A documents our Mathematica code, which generates graphical representations of meaning lists.

II. POLICY AND THE POLICY WORKBENCH

The term *policy* is being used in this paper in the same way that an ordinary person would use it in normal conversation: as set of rules that may, under extreme circumstances, be violated, but which otherwise constrain or delineate the behavior of the different actors within a group or organization. Even so, when used within a software engineering context, the term *policy* is an ephemeral one [Sibley, Michael, Wexelblat, 1992], perhaps because the relationships between policy and formal systems can be somewhat abstract, or perhaps because practical applications in areas such as networking are not always transparent.

In this section, we discuss policy as a way of organizing the behavior of groups of people, and also as a way of controlling the behavior of distributed hardware systems. Two key points are emphasized. First, policy sets contain logical structure that can be analyzed and refined through formal methods akin to those used in software development. Second, policy supports certain abstractions that can significantly simplify the tasks of all policy stakeholders.

A. MOTIVATION: POLICY, COMPLEXITY, AND ABSTRACTIONS

A passenger suffers a heart attack onboard a rush-hour commuter train. The conductor does not stop for medical aid, but instead continues on route, picking up passengers along the way. The unfortunate heart attack victim dies within an hour.

During the subsequent inquiry, a general manager for the transit authority stated that there is a clear policy for medical emergencies onboard a train: stop at the next station and wait for an ambulance. However, an assistant conductor who was at the scene pointed out that the train operates on CSX owned tracks, and that CSX rules forbid trains to make unauthorized stops or to ride through a station without stopping. It was also pointed out that during a previous medical emergency a train stopped at a local station and had to wait 25 minutes for an ambulance; continuing on into the city might have actually reduced the wait for professional medical assistance. A spokesperson for Amtrak said that Amtrak was “very interested” in determining exactly who had jurisdiction over the train and the tracks [The New York Times, 1 August 2002].

As this story illustrates, policy plays a critically important role in shaping the behavior expressed by the individuals within an organization. Policies are pervasive in the civilized world, and whether assuming the role of taxpayer, motorist, or competitor in an economy, we often follow policies without even thinking. It is easy to overlook that policies are usually intended to provide benefits to the group as a whole at the expense of some cost, constraint, or increased effort to each individual.

Yet as the story also illustrates, policy can be poorly designed and justified, difficult to communicate, and even internally inconsistent, and can thus lead to results that benefit no one. These characteristics are symptoms of a more fundamental problem: complexity. To a person required to conform to a policy set, complexity can arise due to coupling between policy statements, conflicts (real and/or apparent) between policy statements, or simply because of large numbers of policy statements. Complexity can also impact other policy stakeholders; for example, it may not be clear to a policymaker how to construct a policy set to achieve a given set of organizational goals. Implicit, unwritten policies compound this complexity, as does the common assumption that exceptions to policy can be made under extreme circumstances. Complexity can thus confound the efforts of all policy stakeholders: policy developers, policy maintainers, policy enforcers, and those who must conform to policy.

Policy is not the only discipline that faces significant boundaries imposed by complexity. It has been known for many years [Brooks, 1995] that complexity limits the scope and sophistication of practical software systems. One of the most important mechanisms used by software engineers to deal with complexity is *abstraction*. As defined in [Berzins and Luqi, 1990, p. 2], an abstraction is a simplified view of a system which contains only the details significant for a particular purpose. Abstractions are essential for the development and maintenance of large-scale, complex systems, because only through abstractions can such systems be analyzed and synthesized by individual persons.

Abstractions play a similar role with policies. It is generally accepted [Michael, Ong, Rowe, 2002] that policies can be broadly classified as meta-policy, goal-oriented policy, and operational policy, and a look at each of these policy types indicates the sort

of support they can provide for abstractions. (The definitions and examples that follow are from [Michael, Ong, Rowe, 2002].) A meta-policy is a statement of policy about policy. Examples of meta-policy include "All passwords must comply with Security Statement XYZ," and "All policies regarding the removal of equipment from a building are security policies." Statements of meta-policy provide simplifying abstractions because they allow policy to be grouped and reused.

A statement of goal-oriented policy specifies only the required final circumstances that the domain object must meet. (A *domain object* is a person or thing that must obey a given policy statement.) As an example, the policy statement "Passwords must be hard to guess" requires domain objects to devise passwords that satisfy this goal; the exact mechanism they use is of no concern. Goal-oriented policies allow (and in fact, require) domain objects to determine for themselves the mechanics of achieving the required goal. Policies of this type free the policymaker to think more abstractly, because the policymaker is not required to explicitly specify how the desired circumstances are to be achieved.

Goal-oriented policies allow the high-level goals of an organization to be decomposed into a hierarchical collection of "miniature" goals. This hierarchy of goals is also a hierarchy of abstractions: the high-level goals are the most abstract, and the lower-level goals are necessarily much less abstract, in the sense that achieving the lower-level goals is normally expected to require less thought or "creativity" on the part of the domain objects at that level.

In contrast, a statement of operational policy is one that specifies explicitly a procedure that must be carried out, or a low-level requirement that must be satisfied. Examples of operational policy statements include "Passwords must be changed every three months," and "Passwords must contain at least one special character." Statements of operational policy certainly incorporate abstractions; in fact, general statements in any natural language express a wealth of information that is not detailed explicitly. However, it could be argued that operational policy is distinguished because, to some extent, it provides domain objects with the exact opposite of an abstraction: operational policies are explicit instructions or tasks that can be carried out in a mechanical way, and the

domain objects are relieved of even minimal judgment, creativity, and responsibility. Operational policies allow a policymaker to use abstractions in an arbitrary way as the goals and high-level policies of an organization are being developed, but to insulate domain objects from those abstractions. This insulation may be of value when the high-level goals of an organization are complex, confidential, or simply of no significance to the domain objects.

B. POLICY-BASED NETWORK-MANAGEMENT

For many of the same reasons that policy is used to manage the behavior of organizations of people, policy is of late emerging as the preferred way of managing distributed computing systems and networks [Sloman, Lobo, Lupu, 2001]. Managing a network of heterogeneous servers, routers, and other components is a difficult task because even a moderately sized network may be impractical to configure manually [Damianou 2001, p. 14]. As clients using a network request, and become willing to pay for, various levels and combinations of security, priority, and performance (that is, various levels of Quality of Service (QoS), in terms of packet loss, delay, jitter, etc.), the complexities associated with network management will increase correspondingly. Hardware and software that allows network managers to specify network behavior in terms of policy is simplifying network control, in large part by abstracting away from the network administrator many of the inconsequential details of network configuration.

One way of configuring a policy-based network-management scheme is shown in Figure 1 [Joyce and Walker, 1999]. A policy server contains a policy repository, which holds a complete database of all policy statements. Network conditions are monitored by the policy server (or more generally, at a Policy Decision Point (PDP)), and when conditions warrant, a command to change configuration is sent to the appropriate elements in the network, such as switches, routers, and firewalls, which are also known as Policy Enforcement Points (PEPs). This command may be sent directly to the network element if the element contains a policy agent to interpret the command. If the network element is not capable of interpreting the command, a stand-alone configuration proxy issues the device specific configuration commands.

The arrows shown in Figure 1 indicate information flows between the policy server and the various policy support and network elements. It is important to note that the flow of information is not one-way; the Policy Server may need to configure the network based on information that is not available at the appropriate network elements. For example, a policy could potentially specify that when traffic becomes heavy at one point in a network, certain actions be taken at another, far-away point in the network. Thus, by sampling the network at a rate somewhat faster than the maximum rate at which its characteristics might change, the policy server can exercise a form of "global" control over the network. This type of control could not be carried out by any individual network element with its locally available information.

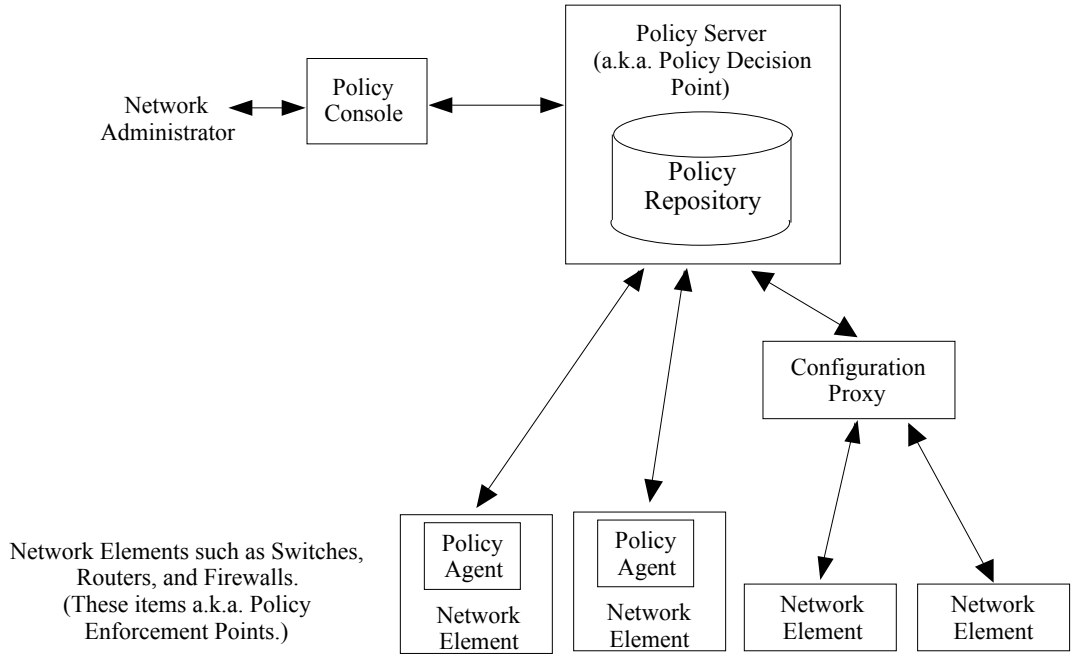

Figure 1. Hardware relationships in a policy-based network-management scheme [Joyce and Walker, 1999]. Arrows indicate communication flows. A Configuration Proxy allows policy-based network-management to be used even with legacy hardware that does not contain a Policy Agent.

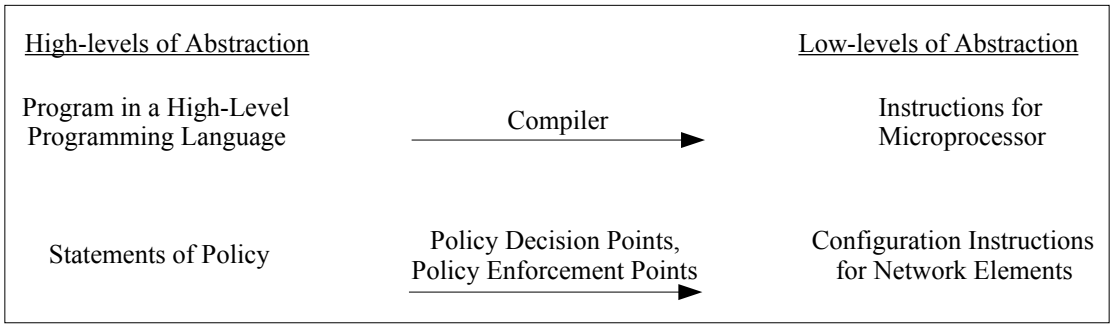

Figure 2. Statements of policy, like programs in a high-level programming language, are representations at a high-level of abstraction. Policy statements apply to distributed systems; that is, to systems composed of large numbers of processing elements. Conventional computer programs are executed on a single processor.

It is important to note that although the use of policy for network configuration provides significant power and flexibility to a network administrator, this power and flexibility can potentially incur a significant cost if complexity is not managed. It is well known that even the most thorough testing of an ordinary desktop application will exercise only a small number of the possible paths through the code, and only a small fraction of the values that the variables can simultaneously take; challenges at least as significant can be expected to hold for policy sets controlling a network. Novel failure modes, such as instabilities in a computer communication network can also arise if policy is not chosen carefully. That is, because a policy server makes policy-related decisions based on the conditions it senses on the network, the behaviors of network elements could potentially switch back and forth between two or more unstable states. Careful analysis of policy sets before deployment on a network could play an important part in avoiding this and other problems.

C. THE RATE AT WHICH POLICIES CHANGE

To avoid confusion when reading the literature, is important to keep in mind that when considering the rate at which policies change, two different time scales must often be considered. A natural time scale to use when considering the action of any policy is that over which the policy applies in a single case of application. For example, a natural time scale for considering traffic laws would be on the order of minutes or hours, which

is the time for a typical automobile trip to which those policies would apply. Similarly, a natural time scale for considering tax laws in the United States would be a year, because tax laws typically apply over that span of time.

For the most part, policies do not change quickly when measured in their "natural" time scale: traffic laws usually change slowly with respect to the pace of traffic, and though each year sees a great number of changes to tax laws, those changes typically represent only a small fraction of the total set of tax laws. In this sense, it is appropriate to say that "...policies are relatively static compared to the state of the managed system" [Damianou 2002, p. 17].

However, even if policies change slowly with respect to the state of the managed system, that change may be very quick in a real-time sense. Furthermore, policies that change slowly with respect to the state of the managed system still require careful consideration by all stakeholders, because any change, even when apparently small with respect to the entire policy set, can potentially influence any other policy. Thus, even when policies are relatively static with respect to the state of the managed system, computer support for policy management can be extremely valuable.

D. THE POLICY WORKBENCH

In addition to supporting abstractions that simplify conceptual and practical work, policy within an organization can be changed or refined relatively quickly to address, for example, a non-stationary environment or to support evolving organizational goals. The important role that policy plays in determining the behavior of an organization, combined with the fact that policy can be changed quickly and cheaply, suggests the following analogy: In a broad sense, an organization behaves under the control of a particular policy set in much the same way that a hardware microprocessor behaves under the control of a particular software program. That is, though a policy set does not comprehensively specify every detail of a human agents behavior (such as how to answer a phone), it does specify an agents behavior in enough detail that successful policies can be distinguished from unsuccessful policies. Furthermore, policy sets typically contain a

large number of policies, these policies are typically interrelated, and policies specify behaviors to a non-trivial amount of detail. These considerations taken together suggest that tools like those used in software development [Berzins and Luqi, 1990] may provide valuable support to various policy stakeholders.

The policy workbench, proposed in [Sibley, Michael, Wexelblat, 1992], is an integrated collection of software tools for aiding in the formal analysis of policy. Among the tasks that the policy workbench is envisioned to perform are the following.

- A policy workbench can check that a set of policy statements are consistent: That is, that no statement conflicts with any other statement, either directly or indirectly. Two types of consistency checks can potentially be carried out: a collection of policy statements can be checked simultaneously ("batch" testing), or a new policy statement can be checked against a set known to be consistent (regression testing).
- Given a real or hypothetical scenario, the policy workbench can retrieve applicable policy statements.
- Alternately, the policy workbench can analyze a given scenario and determine whether policy is being conformed to or violated. This functionality of the policy workbench addresses the problem of communicating policy, and supports both those who must enforce policy as well as those who must conform to it.
- A policy workbench can evaluate requests for exceptions, providing information about an exceptions implications.
- Finally, a policy workbench can maintain version control over policy, so that all stakeholders can be assured of having access to the latest version of policy, and that policy that was in effect at some particular time in the past can be retrieved.

Policies are statements about the allowable behaviors of an organization; however, policies are generally developed to promote the high-level goals of an organization. Because it is often unclear how best to shape the behaviors of individuals

to move towards a set of high-level goals, it makes sense to analyze and refine policy statements with a tool like the policy workbench before imposing them.

E. THE HUMAN-MACHINE INTERFACE FOR A POLICY WORKBENCH

The policy workbench was conceived as a tool for lawmakers and enforcers, business managers, information security personnel, and anyone else involved in the creation, enforcement, and conformance to policy. It was recognized during early work [Sibley, Michael, Wexelblat, 1992] that it would be an unacceptable distraction for most policy stakeholders to learn an artificial language, even one that was somewhat "natural," to interact with a policy workbench.

This provides the motivation for a natural-language input system for a policy workbench, but we see a natural-language input system as only one part of a comprehensive human-machine interface for a policy workbench. For the natural-language input system to be of maximum effectiveness and efficiency, we envision the human-machine interface for a policy workbench as including an output system that provides the user with direct and immediate feedback on how the workbench is interpreting the input. That is, after providing a natural language statement of any sort to the policy workbench, and before the next natural language statement is submitted, the workbench should respond with an expression stating its interpretation of the human input. Ideally, this expression would be in natural language text. However, even feedback consisting of the workbenches interpretation in a computational format would be valuable; this is because the human operator would only need to understand the language, and would not have to generate statements in this language. The purpose of this feedback would be to give the human operator the opportunity to catch mistaken interpretations as soon as possible, and to provide corrections while the intent of the policy was still in mind.

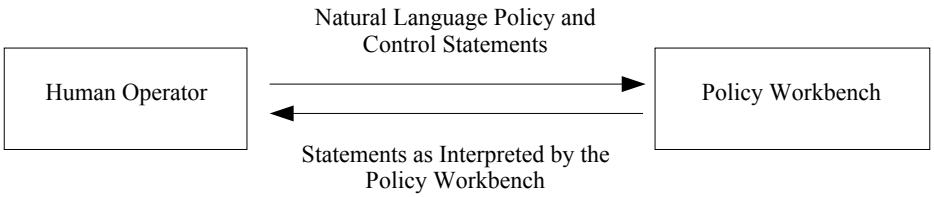

Figure 3. The human operator of a policy workbench provides policy and control statements in a natural-language format, and receives immediate feedback about how the natural-language statements have been interpreted. Though feedback in a graphical or natural-language format would be optimal, such formats are not absolutely necessary.

Direct and immediate feedback to a user, as indicated in Figure 3, will compound the benefits of a natural-language input system. The most important benefit of such feedback is that it would help to maintain the integrity of the policy database: it allows the user to take corrective action as soon as a misinterpretation on the part of the workbench occurs. Without this feedback, inputs that are misstated by the user, or equivalently misinterpreted by the policy workbench, will become integrated within the policy database. The case of the human-machine interface without feedback is similar to that of an open-loop control system, while the case with feedback is analogous to that of a closed-loop control system.

Other benefits can be identified. Information about how an input is being interpreted could allow the user to provide corrective feedback to natural-language processing systems that adaptively learn from their mistakes. Conversely, a policy workbench with a mature and stable natural-language processing system could return information to a user to help them learn about the capabilities of the policy workbench, or even to help them learn certain specifics of a particular application domain.

III. THE POLICY WORKBENCH AND NATURAL-LANGUAGE SUPPORT FOR POLICY: PREVIOUS WORK

In this section, we discuss six papers that present ideas relevant to our later discussions. The paper by [Sibley, Michael, and Wexelblat, 1992] introduces the concept of the policy workbench and explores issues associated with its development and use. The second paper, [Michael, Ong, and Rowe, 2002], describes the development of a prototype natural-language input system for a policy workbench. Our work in Section IV of this paper uses [Michael, Ong, and Rowe, 2002] as a baseline for comparison.

[Cuppens and Saurel, 1996] and [Sergot et. al., 1986] both present studies involving the manual translation of natural language policy specifications into a computational form appropriate for computer-aided analysis; these works are of interest here because we are developing a system to carry out automatically what they have successfully done through manual labor. [Damianou et. al., 2001] describe the policy specification language Ponder, which provides a common language for specifying access and control implementation mechanisms for distributed object systems. [Sloane, 1991] describes the deployment of a system that is in some ways similar to a policy workbench onboard a United States Navy ship.

Finally in this section we discuss some basic concepts associated with natural language processing that will provide support for our later work. We present brief summaries of modal auxiliaries and quantifiers, fuzzy descriptors, speech act theory, and anaphoric references.

A. USE OF AN EXPERIMENTAL POLICY WORKBENCH: DESCRIPTION AND PRELIMINARY RESULTS

The first part of [Sibley, Michael, and Wexelblat, 1992] presents a requirements analysis, a preliminary design, and a prototype implementation for an integrated collection of tools to support the formulation, analysis, and implementation of policy. The set of tools, referred to as a *policy workbench*, is envisioned to be a general purpose system applicable to any policy domain, and requires only that the user have appropriate domain knowledge, with no specialized knowledge in formal systems necessary. The

second part of the paper presents two case studies that illustrate the potential use of a policy workbench.

The paper begins with a requirements analysis for the policy workbench. Several observations, summarized below, are established that are important for any work with policy.

- Policies are written in natural-language, and thus tend to be imprecise.
- Policies are often incomplete or inconsistent.
- Policies are often coupled together; that is, they are interdependent.
- In real-world applications, a policy set can be very large.
- There is often a need to represent *time* in relatively sophisticated ways. They point out that it may be necessary to represent time in absolute terms, such as "...the fiscal year starts at midnight on Sept. 30...", as opposed to just the ordering of events.
- In some cases it may be necessary to represent *intent* of an actor.
- Policies fall along a wide scope of comprehensiveness, from very general to very specific.

The policy workbench itself, shown below in Figure 4, consists primarily of the following three components.

- *Theorem and Assertion Analyzer* This component of the policy workbench verifies that inputs are syntactically correct, and that certain semantic conditions are satisfied, such as consistency with existing policy. If an input is found to be syntactically correct and consistent with existing policy statements, the policy database is updated. This component is also responsible for configuration management of the policy database.
- *Compiler-Generator-Interpreter* This component is used to determine the consistency of procedures and scenarios with the statements in the policy database. That is, it allows a user to ask "what if?" type questions of the policy

set. The user generates code representing a procedure or scenario using a computer-aided software engineering (CASE) tool. This code is merged with pre- and post-conditions which represent the structure of the policy statements. The resulting software can be analyzed to determine whether any solutions exist, or whether inconsistencies have occurred.

- *Policy Selector* The purpose of this component is to extract subsets of policy from the policy database for the user. Though work had just begun on this component when [Sibley, Michael, and Wexelblat, 1992] was written, an analysis of pre- and post-conditions associated with statements in the policy database was envisioned to allow commonalities in policy statements to be determined.

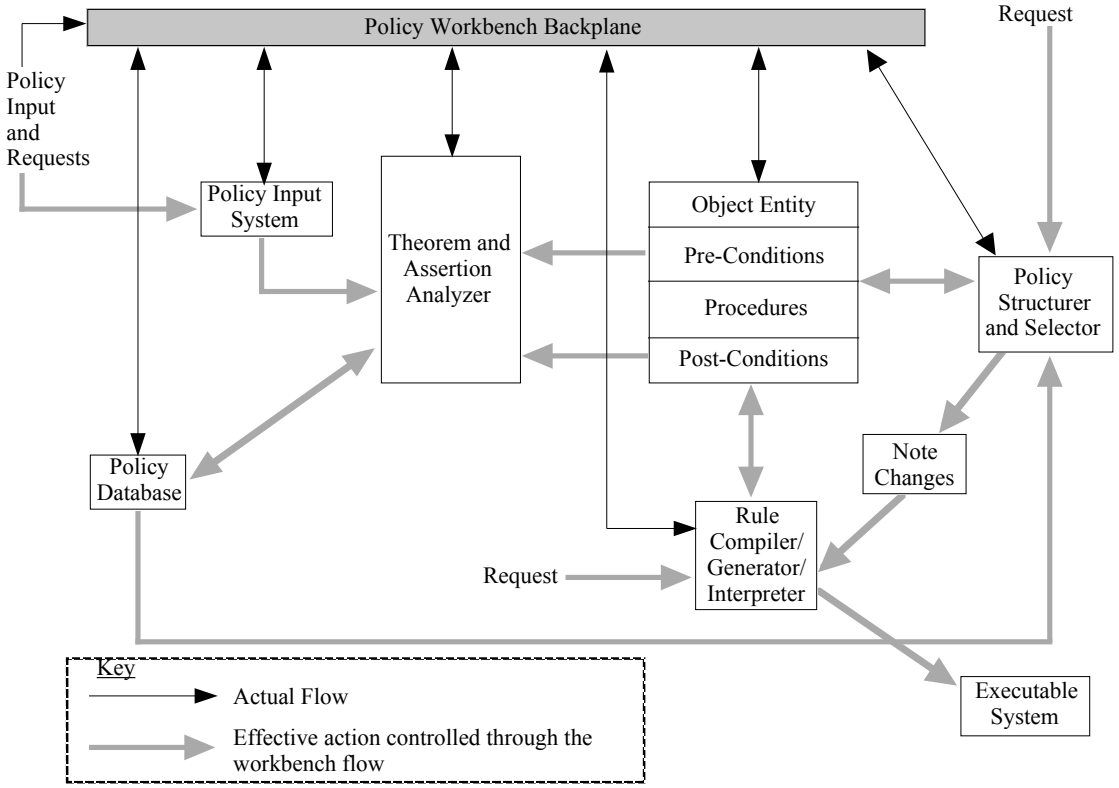

Figure 4. Relationships between policy workbench tools. Adapted from [Sibley, Michael, Wexelblat, 1992, Fig. 1].

To illustrate the use of a policy workbench, a case study involving the analysis of a set of security policies for a secure work area is presented. A set of 21 policy

statements, 11 real-world facts, and nine statements from a hypothetical "employee manual" were analyzed, converted into a computational form, and processed with a theorem prover. The preliminary policy analysis phase required very little computer support, because it involved defining the problem and establishing the real-world facts needed for subsequent analysis to be automated. However, checking for inconsistencies, determining whether a policy set was minimal in a formal logic sense, and generating the logical consequences of the policy database relied heavily on OTTER, a resolution-style theorem prover program. The authors report that the time required for analysis by the theorem prover was sometimes significant, and that techniques for reducing the size of the search space were thus an important area for future research.

B. NATURAL-LANGUAGE PROCESSING SUPPORT FOR DEVELOPING POLICY-GOVERNED SOFTWARE SYSTEMS

[Michael, Ong, and Rowe, 2001] describe the architecture of a Natural-Language Input Processing Tool (NLIPT), and report experiments that they performed with a prototype of one component of it, the extractor. Their NLIPT, shown below in Figure 5, is comprehensive in the sense that it processes all user input, including policy statements, queries, and scenarios. The NLIPT operates by generating a meaning list representation of the natural-language input, and then by generating a set of key index terms by selecting and weighting terms from the meaning list. The key index terms are used to retrieve one or more structural schema of applicable policy statements from the Policy Element Identifier Tool of the policy workbench. The structural schema returned by the policy element identifier tool contain implicit facts and hierarchical relationships that can be exploited by the structural modeler to generate a conceptual schema from the meaning list. The conceptual schema is then processed by the logic modeler, which infers such things as quantifiers and their scope, to generate a first-order predicate logic representation of the input.

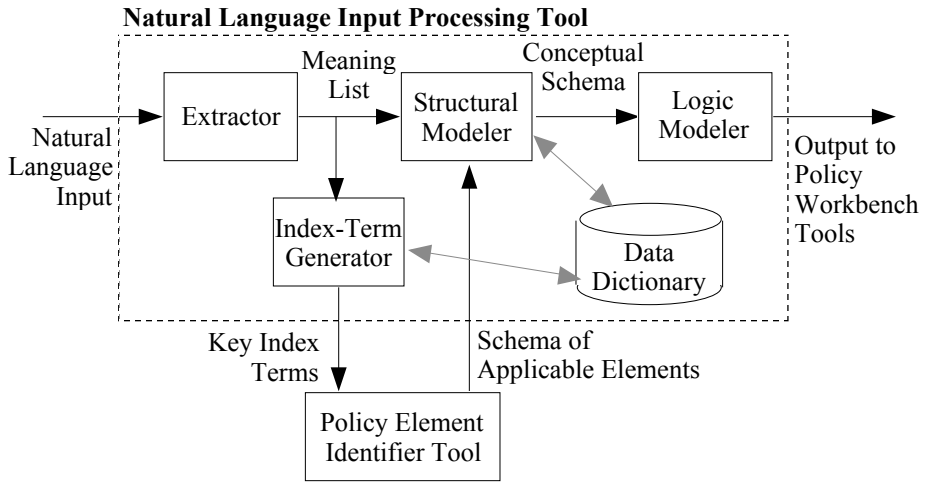

Figure 5. Architecture of the Natural-Language Input Processing Tool. Figure adapted from [Michael, Ong, and Rowe, 2001, Figure 1].

The prototype extractor operates, as shown in Figure 6, by generating a part-of-speech tag for each word in the natural-language input, and by grouping together, or "chunking," multiword syntactical units such as noun and verb phrases. This tagged and chunked input is converted into a series of Prolog predicates by an intermediate processor written in Java. A program written in Prolog then generates a meaning list for the input using a finite state grammar that was tailored to capitalize on the structure expected to be seen in typical policy statements.

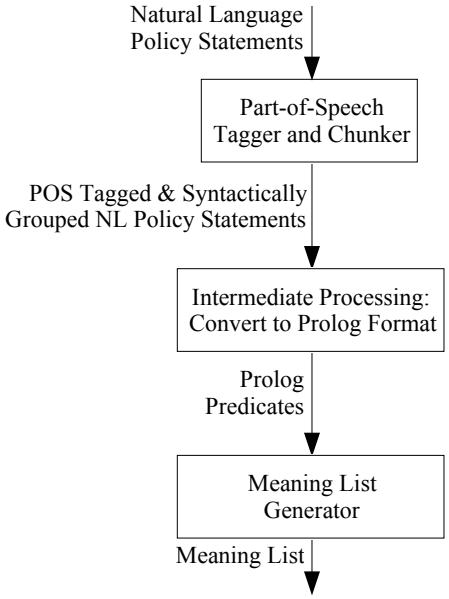

Figure 6. Data flow diagram for the Extractor. Adapted from [Ong, 2001, p. 41, Fig. 5].

The prototype extractor was tested by applying it to a collection of policy statements for a university web site, and it was found that 96% of the meaning list terms it generated were correct. Full sentences consisted, on the average, of eighteen meaning list facts, so that the number of wrong meaning list facts was about (1-0.96)*18=0.72 per sentence; in other words, there was more than one error per two sentences. The authors point out that the use of a full context-free grammar in the meaning list generator would likely improve performance, as would refinements to the part-of-speech tagger and chunker.

C. SPECIFYING A SECURITY POLICY: A CASE STUDY

The work presented in [Cuppens and Saurel, 1996] provides significant insights into the relationships between natural language statements of policy and equivalent computational forms. The authors present a logic-based language for expressing statements from a natural-language security policy, and they use a case study to both motivate key components of the language and to illustrate how the language is applied. A security document consisting of 60 pages of natural-language text was studied, and the main logical concepts used within it were identified and then formalized in the (unnamed) language. The authors state that the expressiveness of the security document was rich, and it expressed concepts such as obligation, permission, prohibition, responsibility, and delegation.

The language developed in [Cuppens and Saurel, 1996] is intended to allow security administrators to specify, define and formalize security policies for a particular high-security-risk environment. Though the case study presented in this paper focuses only on the formalization of security policies, the language being developed is envisioned to provide representations that will allow security administrators to carry out the following tasks.

- Query a given security policy;
- Verify that properties such as consistency and completeness are enforced by a given policy;

- Verify that a given situation does not violate the security policy; and
- Investigate interoperability problems between several security policies.

To represent policy statements like those found in the security document, the authors incorporate into their language the fundamental constructs of objects, events, actions, and agents. Additional concepts such as delegation, obligation, permission, and prohibition were also included. Concepts such as obligation, permission, and prohibition are characteristic of deontic logic, which is used to reason about ideal and actual behaviors [Meyer et. al., 1996]. Deontic logic is an important component of the language presented in this paper.

The language uses objects as a way of organizing data and associated operations, and also to support inheritance so that different levels of abstraction can be represented in a hierarchical way. Events are used to express ways in which the environment may change. Actions are applied to objects that occur in a regulation, and are expressed as methods of an object. Both events and actions have temporal constructs associated with them: the *Event* class has the boolean attributes *Before*, *During*, and *After*, and actions support the three corresponding binary predicates *Before_Exec*, *During_Exec*, and *Exec* (which is an abbreviation for *After_Exec*).

The language also provides constructs for further deontic expressions such as obligation, permission, prohibition, and for actions that are often implicit or indirect. For example, a goal-oriented policy will only specify the results of an action without specifying the actions themselves. Similarly, a policy statement may only specify the person that is responsible for ensuring that an action is carried out; the agent who actually performs the action is not named. To support implicit and indirect actions such as these, the language provides the operator *Do*. Informally, if *a* is an agent, and *p* is a formula that describes an expected effect of an action, the expression *Do(a,p)* specifies that *a* brings it about, or sees to it, that *p* is the case.

The authors point out that many of the statements from the policy document can be put in the form: if some condition is satisfied, then a particular agent is prohibited, obligated, or permitted to do something. This point is supported by several examples

which show natural language statements taken from the policy document along with the corresponding computational representation in their formal language. The following is one of these examples [Cuppens and Saurel, 1996, p. 131].

During the time an agent is elaborating a document classified at the secret level, he/she is obliged to work in a protected area.

```
  Agent(a)
∧ Classified_Document(d)
∧ Classification(d)=Secret
∧ During_Exec(a,Elaborated(d))
∧  O ∃z, (Protected_Area(z)
      ∧During_Exec(a, Work(z)))
```

This example illustrates the use of the deontic modality "**O**," which specifies an obligation. Except for a few policy statements with particularly complex structures, the authors were successful in representing all statements from the policy document in their formal language.

Of particular interest are the difficulties the authors encountered in interpreting and translating the natural language statements from the security document. The authors note that a significant number of these statements could only be interpreted as simple observations of fact, or statements of advice; such statements were discarded. Another difficulty was "... to identify which agents are concerned with each norm of the instruction: this information was often left implicit in the regulation." That is, the natural language policy statements often left unspecified the particular domain objects that the policy applied to. A further difficulty was the intentional ambiguity in the subject policy document: "the lack of precision ... was deliberate in order to make the regulation applicable by a large variety of organizations."

D. THE BRITISH NATIONALITY ACT AS A LOGIC PROGRAM

As with [Cuppens and Saurel, 1996], our interest in [Sergot et. al., 1986] comes about (in part) because they have done "by hand" what we hope to accomplish automatically: they have successfully converted a piece of statutory law, the British Nationality Act of 1981, into a logical representation appropriate for computer-aided

analysis. The work of [Sergot et. al., 1986] is also noteworthy for its observations on logic programming (especially the formalization of negation), and applications of formal representations.

[Sergot et. al., 1986] generate their representation of the British Nationality Act in terms of Horn-clause logic, a segment of first-order logic which, though restricted in its expressiveness, lends itself well to efficient theorem provers. Their formalization was developed in a top-down way. That is, high-level concepts (for example, "place of settlement") were successively replaced with lower level concepts until reduced to undefined, concrete concepts, which can in turn were established by reference to the particular case under study, and through various other means.

Their development also occurred with a significant amount of trial and error: logical representations for a particular clause would be generated, and then refined in accord with observed shortcomings. According to standard programming practices, this approach would be deemed bad practice; good methodology requires a correct and complete program specification before coding begins. However, the authors point out that the formalization of legislation and policy is much more like the development of a formal specification for a piece of software, or like the development of axioms for an axiomatic mathematical system, than it is to the development of a program. That is, though their finished product has the outward appearance of a program, in substance it is much more like a formal specification; they essentially converted an informal specification into a formal specification.

E. PONDER: A LANGUAGE FOR POLICY SPECIFICATION

Though technologies like active networks, mobile agents, and management by delegation are increasing the flexibility and functionality of network services, they are also compounding the magnitude and complexity of tasks associated with maintaining a network and ensuring the integrity of the information traversing it. Policy-based network management is one approach being developed for dealing with these problems. Policy-based network management is based on a strict separation of network implementation from the policies used to manage it, so that the policy-based description of network

behavior can be changed without requiring changes to the underlying network. Ponder [Damianou et. al., 2001] is a policy-specification language developed to support policy-based management of the various entities that make up distributed-object systems.

The policy types that Ponder supports include access control policies, obligation policies, and constraints. Access control policies deal with limiting the activities of legitimate users (that is, users who have been successfully authenticated). Ponder supports access control through positive and negative authorization policies, information filtering policies, delegation policies, and refrain policies. Ponder also supports obligation policies, which specify actions that must be performed by certain actors when certain events occur.

A variety of techniques are available in Ponder for simplifying and organizing policy. Constraints can be expressed in Ponder through a subset of the Object Constraint Language, and policy can be specified in terms of roles and groups. Specifying policies in terms of roles, instead of specific individuals permits personnel to be changed without requiring that policy statements be changed, while groups allow policy sets to reflect organizational structure, providing simplicity and opportunities for reuse.

A significant concept used with Ponder is that of domains. Domains are a means of partitioning objects to which policies apply according to geographic boundaries, object type, or other ways convenient for human managers. Domains are similar to directories, and are implemented through the use of LDAP services.

F. THE USE OF ARTIFICIAL INTELLIGENCE BY THE UNITED STATES NAVY

[Sloane, 1991] describes how a sophisticated software system for knowledge extraction, storage, and transfer was introduced on board a nuclear-powered aircraft carrier. This software system, despite its potential, never "caught-on" with ship personnel because it was perceived as not supporting users needs. The system eventually died from neglect. A look at some of the reasons behind this failure illustrates how a software tool for improving the performance of an organization can fail for a number of reasons other than its technical quality. In particular, a tool for knowledge storage and

retrieval has to be positioned within an organization so that its capabilities are relevant and useful to end users. Also, users must be provided with appropriate resources, such as time and training, to successfully exploit the tool.

The deployed software system, called "ZOG," was conceived to provide benefits similar to those that would be provided by a simple policy workbench. Specifically, it was to allow ship personnel to extract information about billets, individuals, and task responsibilities as the need for such information arose during problem solving. A critical aspect of ZOG was that the information within it was to come from the crew itself: ZOG's information databases were to be populated and constantly refined by the ships officers and crew, so that knowledge gained through experience would be retained despite personnel turnover. ZOG would thus become a hardcopy record of corporate procedures, memory, and expertise.

For a number of reasons, however, ZOG was rarely consulted, and even less frequently updated. In part this was because the ship was first and foremost required to perform as an operational aircraft carrier: Operational readiness was of the utmost concern for shipboard personnel, and as a research project with benefits that were not immediately visible, consulting and maintaining ZOG became a low priority.

Of equal significance was the problem of responsibility assignment. A deployed aircraft carrier is among the most hazardous work environment in the world, and as a consequence the ships officers and crew bear a heavy responsibility for all decisions and actions. This responsibility could not be assumed by a machine, and so ZOG's output was reduced in the eyes of users from credible guidance to mere suggestions requiring evaluation by the user.

The pace of onboard decisions and actions also served to marginalize ZOG. In many cases it was critical not just that the right decision be made, but also that the right decision be made quickly. ZOG's keyboard driven menu system and the uncertain quality of its outputs made ZOG too inefficient for practical use.

G. NOTES ON NATURAL LANGUAGE PROCESSING TECHNOLOGY

1. Modal Auxiliaries and Quantifiers

Modal auxiliaries appear frequently in natural language statements of policy, and they are important because they mark deontic logic constructs common in policy statements. As explained in [Hodges and Whitten, 1982], an auxiliary verb is a verb used with a main verb in a verb phrase; examples include *be*, *have* and *do*. An auxiliary typically indicates tense, and may also indicate voice, mood, person, and/or number. Modal auxiliaries are auxiliaries that do not take inflectional endings such as *-s*, *-ing*, or *-en*. Examples of modal auxiliaries include *will*, *would*, *shall*, *should*, *may*, *might*, *must*, *can*, and *could*.

It is interesting to note that "there is only partial agreement among linguists as to which words are modal auxiliaries" [McCawley, 1998], though there are several linguistic requirements that at least some linguists hold to be necessary for something to be a modal. These requirements are for the most part quite technical, but one states that a "modal" term is one that refers to "alternate possible worlds" by expressing a notion such as possibility, necessity, or desirability.

As explained in [Jurafsky and Martin, 2000], a number of different kinds of word classes can appear in a noun phrase between a determiner and the head noun; these include cardinal numbers, ordinal numbers, and quantifiers. A quantifier specifies for how much or for which part of a domain a propositional function is true. For example [example from McCawley, 1998], in "Some politicians are honest," the quantifier "some" indicates that "x is honest" is true of some non-empty part of the domain defined by "politician." Stated in another way, all politicians are either honest or not honest, and those that are honest make up some non-zero fraction of all politicians.

2. Fuzzy Descriptors and Their Modeling

In some cases an element of a domain may not make a proposition completely true or completely false; sometimes an element of a domain makes a proposition true only in some qualified sense. For example, when stating whether a particular individual

fits a descriptor such as “fat” or “jolly,” the answer in most cases is not a clear and simple “yes” or “no,” but rather a more qualified phrase like “sort of.” [McCawley, 1993] In a similar way, a natural language policy statement for a web site might refer to a “large graphic,” or might require that a certain HTML construct “be used sparingly,” where “large” and “sparingly” are not specified in detail. Using only our intuition, we may feel that the degree to which a given graphic is large is “not really,” “sort of,” or “quite.”

One approach for dealing with these vague terms is through fuzzy logic. As a generalization of normal two-valued logic, fuzzy logic allows predicates to take truth values that fall within the interval [0,1]. For example, we may feel that a particular individual is “jolly” to a degree of 0.7. Similarly, a graphic appearing on a web page may fit the quantifier “large” to the extent 0.4. It is important to note that we are not saying that “the graphic under consideration has a probability of 0.4 of being large.” Rather, we are saying that it fits, or conforms to, the quantifier “large” to the degree 0.4. Something that perfectly fit the term “large” in an unqualified way would be “large” to degree 1.0, while something that absolutely did not fit would be “large” to degree 0.0. Fuzzy descriptors are derived from the concepts of fuzzy logic, and provide a non-probabilistic way to model vague quantities that appear in natural language text. Verbs, nouns, adjectives, and grammatical quantifiers can be fuzzy in natural language text.

Multiple fuzzy terms that appear in a single sentence can be combined according to the formulas (shown in prefix notation) below . These formulas are natural generalizations of the rules for normal 0-1 logic [McCawley, 1993]. However, it is important to note that these equations provide only upper bounds for the terms in question, and these upper bounds can be far from the average case.

$$
\begin{aligned}
/\sim A/ &= 1 - /A/ \\
/\wedge AB/ &= \min(/A/, /B/) \\
/\ \ AB/ &= \max(/A/, /B/) \\
/\ \ AB/ &= 1 \text{ if } /A/ \leq /B/ \\
&= /B/ \text{ if } /A/ > /B/
\end{aligned}
$$

To assign a fuzzy value to an expression, a number of different techniques can be used. Taking as an example the case of determining whether a graphic is “large,” we

might proceed by finding the distribution of graphic sizes within one or more web sites, and then assigning a given graphic a fuzzy value for "large" depending on where it falls in the distribution. In other words, suppose we knew that 80% of the graphic images in some comprehensive sample were smaller in size than the graphic in question. We would assign that graphic a fuzzy value of 0.8 for the attribute "large." We could conclude that a policy statement "Large graphics are prohibited on pages that satisfy X" would be violated to the degree 0.8 by the presence of this graphic on a page that satisfied the criteria X.

3. Speech Act Theory

In the analysis of natural language text, and in the development of software for processing such text, it is important to recognize that there is much more information contained in a natural language expression than is apparent through purely logical means: speech is a sophisticated form of human interaction, and social forces impose a number of conventions and subtleties that must be recognized and understood before information can be successfully extracted. Any speech statement is a social act, just like shaking hands, and is full of subtle conventions that prevent meaning from being extracted through purely logical analysis. Speech has to be interpreted as a social interaction in order to be fully understood [McCawley, 1993]. This is especially important for natural-language policy, where statements may be worded in polite ways that do not appear as imperatives syntactically.

A simple example of a speech act is the question: "Could you close the door?" Though a response of "Yes" would be a logically correct answer, such a response would strike any fluent speaker of English as odd, because the question is in fact a polite request that the door be closed. The speaker knows that the person to whom the question is addressed is capable of closing the door, and is simply asking them to do so in a polite, informal way.

As another example, the following hypothetical letter-of-recommendation [Pinker, 1994] contains only the most positive expressions about the person being

recommended, yet it is unlikely that this letter will gain that person much credibility in the eyes of the recipient. This simple example illustrates that speech acts can appear in written just as well as spoken form.

> Dear Professor Pinker,
>
> I am very pleased to be able to recommend Irving Smith to you. Mr. Smith is a model student. He dresses well and is extremely punctual. I have known Mr. Smith for three years now, and in every way I have found him to be most cooperative. His wife is charming.
>
> Sincerely,
> Professor John Jones

In a literal sense this letter makes only the most positive statements about Mr. Smith, yet at the same time it clearly conveys the message "Stay away from Smith: he's dumb as a tree" [Pinker, 1994]. This twofold meaning comes about because the positive statements about Mr. Smith concern things that are entirely irrelevant in the academic sphere. The author of the letter does not explicitly express his true feelings because an explicit expression of these feelings would be distasteful to both the writer and the reader, and furthermore the author probably wants to distance himself from anything that might cause harm to someone with whom he has a relationship of trust.

Speech acts have no doubt evolved for a number of reasons: to provide an "escape" that prevents people from being embarrassed (as in the example of the letter above), to save effort on the part of both the speaker and listener, and in general to make the communication process more effective and efficient. One consequence of speech being effective and efficient is that "... listeners tacitly expect speakers to be informative, truthful, relevant, clear, unambiguous, brief, and orderly" [Pinker, 1994]. These expectations can provide important information for converting natural language text into a computational form.

4. Anaphoric References

Another complication in interpreting natural language text comes about through anaphoric references. A reference to something (i.e., to a person, place, or thing) that has been previously introduced into the discourse is called *anaphora*, and the referring expression used is said to be *anaphoric* [Jurafsky and Martin, 2000]. In many cases

pronouns such as *he* and *it* are anaphoric references: if someone says, for example, *Ask that policeman, and he will tell you*, the pronoun *he* is a substitute referring to the recently mentioned policeman [Bloomfield, 1984]. However, anaphora can occur through a number of other syntactic structures, including both definite and indefinite noun phrases, demonstratives, and a structure referred to as one-anaphora [Jurafsky and Martin, 2000].

The problem of resolving anaphoric references is significant for processing natural-language policy statements, because often many separate but interrelated statements have to be coordinated into a conceptually unified whole. Fortunately, several algorithms exist for identifying the referents of anaphoric expressions [Jurafsky and Martin, 2000]. One algorithm, due to [Lappin and Leass, 1994], uses weighting factors assigned to seven salience factors to determine the referent for anaphoric references. It consists of five relatively simple steps, and takes into account potential referents that occur up to four sentences back.

A tree-search algorithm [Hobbs, 1978] takes the syntactic representations of previous sentences, and searches for the antecedent noun phrase among these syntax trees. For correct operation, this algorithm depends on having correct and complete syntax trees for previous sentences. An algorithm by [Brennan et. al., 1987] is based on the idea that at any point in a normal dialog, there is a single entity being "centered" on, or in other words that the dialog is focusing on some particular noun phrase. This algorithm attempts to track the entity being centered on.

IV. NATURAL-LANGUAGE SUPPORT FOR POLICY: ARCHITECTURE AND CASE STUDY

The current state-of-the-art in natural-language processing (i.e., [Allen, 1995], [Guglielmo and Rowe, 1992]), along with previous work towards the development of natural-language input systems for a policy workbench ([Michael, Ong, Rowe, 2002], [Ong, 2001]) together support the hypothesis that a robust natural-language input interface for a policy workbench is technically feasible. However, despite the many applications that would benefit from a complete and robust policy workbench with a friendly, easy to use interface, progress towards this goal has been slow. It appears that complexity in components such as the index term generator, the policy element identifier tool, and the structural modeler has played at least some part in slowing this progress. It is well known that complexity is one of the most significant obstacles in the development of sophisticated software systems ([Berzins and Luqi, 1991], [Brooks, 1995]), and that the appropriate use of abstractions, and the appropriate use of geometric or graphical techniques for visualizing data and algorithms can have far reaching benefits. This is especially true in the development of prototype systems.

In this section we argue that the development of a natural-language input processor for a policy workbench can be simplified by representing policies and domain objects as semantic networks, and by exploiting graphical representations of the associated data and algorithms. We begin by reviewing key requirements for a policy workbench, and we then present an architecture based on the use of semantic networks that we anticipate can meet those requirements. We discuss the types of errors that our architecture may commit, and a taxonomy for policy statements. We then present a case study in which ambiguities are identified in a collection of "real-world" policy statements, and transformation rules are proposed for eliminating these ambiguities.]

A. REQUIREMENTS: THE ARCHITECTURE AND ITS OPERATION

In this and the following subsections, we develop, analyze, and refine an architecture to support natural-language input to a policy workbench. Throughout this

process, requirements for our natural-language input processing tool (NLIPT) will include the following.

Our primary design consideration is to reduce the complexity of the natural-language input system, while not reducing the functionality of the workbench itself. Complexity, of course, is a subjective term; our more specific goal is to use semantic networks as a vehicle for introducing geometric structures for the representation of data and for the interpretation of operations carried out on that data. This requirement is motivated by our desire to have in hand a system that can be easily refined, and which has at least sub-components that can be analyzed to some extent.

Ideally, our NLIPT will be able to accept arbitrary statements of natural-language policy and represent them correctly in a computational form. However, it is unlikely that we will be able to develop an ideal system, considering that even humans are well capable of misinterpreting each other. We set for ourselves the "straw-man" requirement that our NLIPT accept arbitrary natural-language input, but falling short of that goal, we prefer a system that accepts natural-language statements that are somehow restricted in scope or structure, rather than a system that requires input based on a formal syntax, even if that syntax is "simple" or "natural" in some sense.

It is to be expected that the NLIPT for a policy workbench will be domain dependent to some extent. The primary reason for this is efficiency: within a particular domain, there are fewer potential meanings associated with a given word in an input statement. Also, probabilities associated with a restricted domain will be more accurate and easier to obtain. However, we set as a requirement that the architecture of the system be as domain independent as practical, and that configuration of the system for a particular domain at runtime require only minimal effort. We envision a system that could be configured for a particular domain by a developer, and for the moment we leave open the question of whether the system could be refined (either automatically, by the user, or by a developer) as it is being used in a particular domain.

Our approach to meeting these requirements is to use semantic networks to represent both the meaning of natural-language policy statements, and particular

instances of systems in the domain of the policy set. As discussed in [Russell and Norvig, 1995], semantic networks are logical reasoning systems that incorporate nodes and relationships in a graph-like structure. A key advantage of semantic networks is that they can be represented in a graphical format, which can help clarify characteristics that might be obscure when presented in a text-based format. Conformance to, or violation of a policy statement by a domain object is established by determining whether the semantic network describing a policy statement is a sub-graph of the semantic network describing the object in the policy's domain. The material presented here reflects our belief that the full power of the first order predicate calculus as a representation language (as used in [Ong, 2001] and [Michael, Ong, Rowe, 2002]) may not be required to get useful functionality out of a policy workbench. The semantic networks used in the following work are less sophisticated than the first-order predicate calculus in the sense that they provide no mechanism for universal quantification, yet at the same time they appear hold the representational power required for useful functionality. This less comprehensive representation provided by semantic networks, along with the fact that they can be represented in an easy to visualize graphical format, promises to circumvent some of the complexities that appear to have held back work based on the full first-order predicate calculus.

A preliminary architecture is illustrated in Figure 6 below. This architecture is considered to be preliminary because the details of several of its components, in particular both of the Semantic Network Generators and the Background & Common Sense Information source, will be specified more fully after the analysis of several policy statements in Subsection IV D. In this subsection we focus only on what the various architectural components are required to do, and not how they are supposed to do it. Also, to make our discussion more concrete we do not present our architecture in its full generality; rather we present our concepts in terms of a system that processes natural-language policy statements for web sites. Generalizations that are not obvious will be explicitly discussed.

Our natural-language input processing tool converts natural-language statements into semantic networks. To illustrate how information in a semantic network format can

be processed, we describe not just the natural-language input processing system, but also a computational technique for determining whether a domain object conforms to or violates a given set of policies. In our more concrete discussions, this translates into a system that determines whether a given web site conforms to the policy contained in a natural-language web policy document.

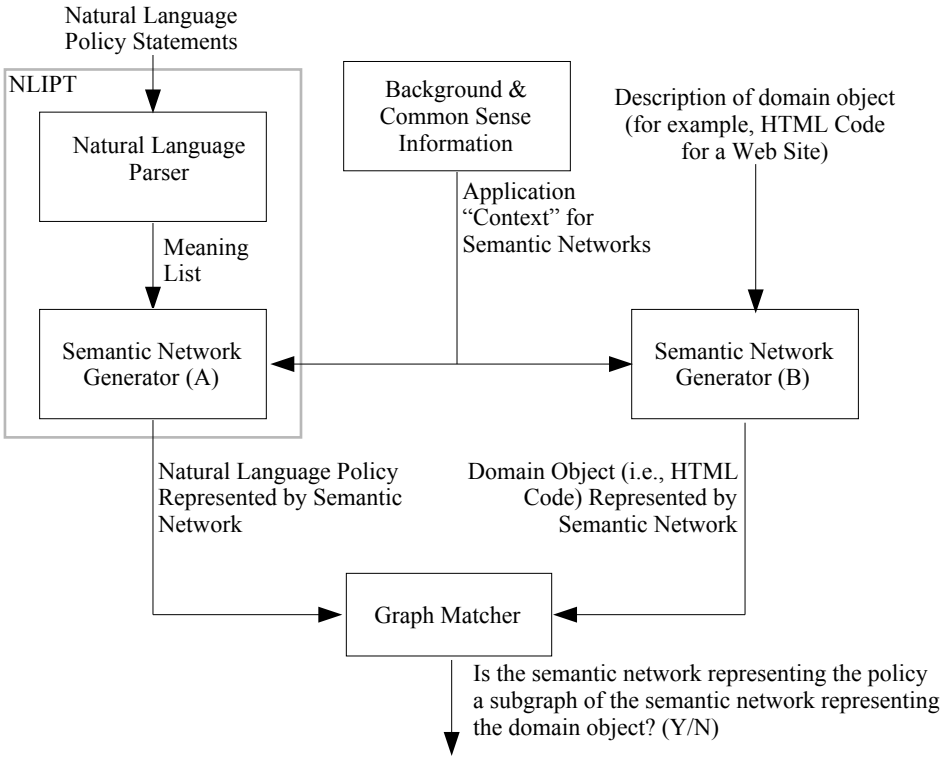

Figure 7. Proposed architecture for a system that automatically determines whether an arbitrary web site conforms to an arbitrary natural-language policy set. The Natural-Language Input Processing Tool is at left in the grey box.

As shown in Figure 7, our architecture consists of several components of the following types.

- *Natural-Language Parser* A parser determines the structure of the input sentence with respect to the rules of a formal grammar [Allen 1995]. The parsing process results in a parse tree and a meaning list. The parse tree describes the structure of the input sentence with respect to the formal grammar, and the meaning list is a representation of the semantic structure of the input sentence. The meaning list

distinguishes the correct meaning and sense of each word used in the input sentence.

- *Semantic Network Generators* Each of the two semantic network generators produces a semantic network. One semantic network represents the set of natural-language policy statements; the other semantic network represents the domain entity. In both cases, input from a *Background and Common Sense Information* source provides supporting information for the development of the semantic networks. The semantic networks could be implemented in any one of several forms; for example, as sets of Prolog statements. The semantic networks for both the natural-language statements of policy, as well as the system to be analyzed, are envisioned to be generated at runtime; neither is to require any off-line processing. However, the Semantic Network Generators themselves will be domain dependent.
- *Graph Matcher* The graph matcher determines whether an isomorphism, or "matching," exists between the semantic network representing a policy statement, and a subset of the semantic network representing the domain object.

The system operates by generating two semantic networks: one to represent a natural-language statement of policy, and another to represent one or more domain objects. The semantic network representation of a natural-language statement of policy is generated from the meaning list produced by a natural-language parser. Similarly, a separate semantic network is generated to represent some number of objects in the domain of the policy statement. To generate this semantic network, a grammar and parser for domain objects will be developed.

The two semantic networks are fed into the Graph Matcher. The Graph Matcher determines whether a semantic network representing a policy statement is a sub-graph of the semantic network representing the domain object. This is similar to, but not identical to, the problem of determining whether an isomorphism exists between two arbitrary graphs. The problem of determining whether an isomorphism, or "matching," exists between two given graphs is very expensive to solve in general, but there exist techniques

that typically work well in practice. As pointed out by [Johnsonbaugh, 1984], "... although every known algorithm to test whether two graphs are isomorphic requires exponential or factorial time in the worst case, there are algorithms that can determine whether 'most' pairs of graphs are isomorphic in linear time..." A low cost algorithm for decomposing our problem (that of determining whether an isomorphism exists between a graph and a subset of another graph) into the more broadly studied problem(that of determining whether an isomorphism exists between two arbitrary graphs) will not be investigated here, but will instead be left as a topic for future work.

We can gain a more solid understanding of this architecture by considering a simple example. Suppose we wanted to determine whether a web site conformed to the natural-language policy statement "Every web site must contain a link to the web-masters e-mail address, so that users can easily report broken links." This statement, minus the superfluous phrase "so that users can easily report broken links," can be represented by the semantic network shown below in Figure 8.

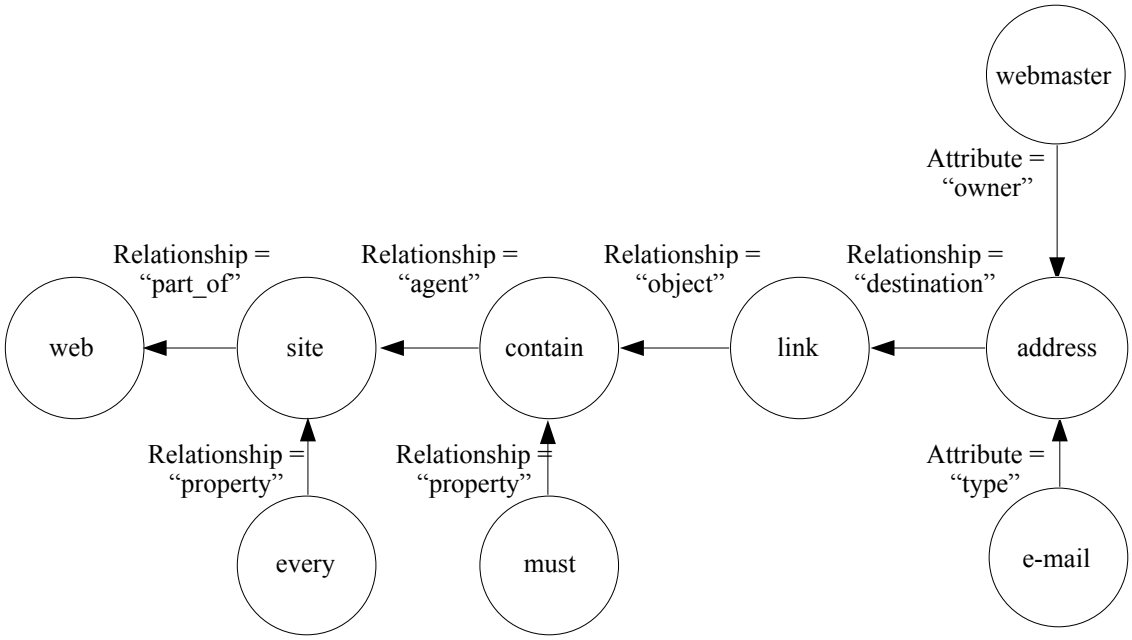

Figure 8. A semantic network representation for the policy statement "Every web site must hold a link to the web-masters e-mail address." Arrows point from the second term in a binary relationship to the first, or from an attribute to the entity that holds it.

To determine whether a web site conforms to, or violates, this policy statement, the graph matcher determines whether the semantic network shown in Figure 8 can be superimposed on, or "matched" with a semantic network representation of the domain object in question, which in this case is a web site. A semantic network representation for a web site might look as shown in Figure 9. The grey line in Figure 9 encloses a semantic network that corresponds to the semantic network representation of the policy, as shown in Figure 8; as a consequence, it can be concluded that the web site conforms to the policy statement.

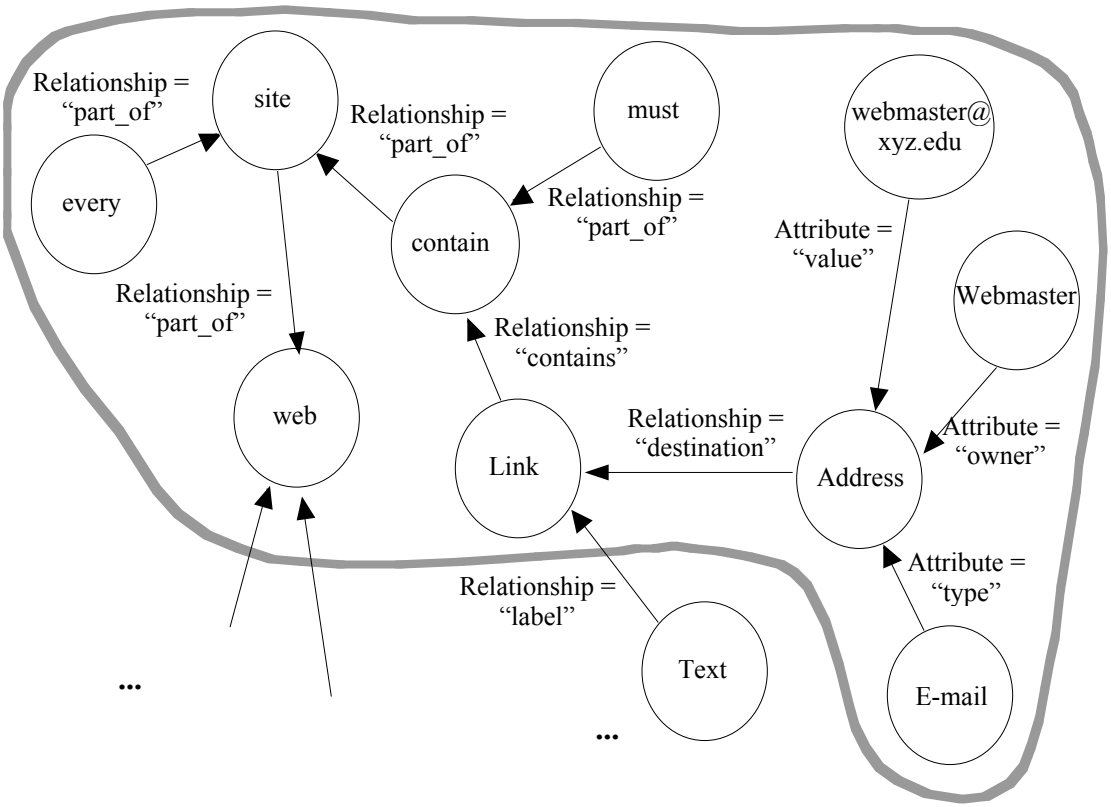

Figure 9. Partial view of a semantic network representation for a hypothetical web site. The semantic network shown in Figure 8 could be superimposed over the sub-network within the grey loop.

This example prompts the following observations.

- Depending on the domain to which the policy statement applies, it may be necessary to perform multiple matches. For example, if the example statement

were changed to read "Every web page must contain a link to the web-master," the domain would consist of all the web pages at the site in question, and each page would have to be checked separately. Searching for the existence of a single item is easier than ensuring that a condition holds in all cases.

- Certain characteristics of the semantic networks have to be standardized, so that insignificant differences in terminology do not prevent a match from occurring. In our example, a web page may "contain" a link, or may "hold" a link; we do not want insignificant differences like this to prevent a match from occurring. This and other problems with synonyms can be dealt with by replacing words with a group of synonyms analogous to the "synonym sets" used in WordNet. For example, we might stipulate that any word in the set {contain, hold} be changed to "hold."

In addition, some key issues involving the generation of the semantic networks must be considered. The Semantic Network Generators are responsible for producing the formal representations of both natural-language policy statements, and of domain objects. In general, producing a formal semantic network representation for a domain object may be as difficult as generating a computational form for natural-language, though it is unlikely to be more difficult. The case of a web site is expected to be relatively simple, because HTML is already a formal representation and thus only the structure of the statements has to be changed. The time required for current natural-language parsers to generate a parse tree and a meaning list is on the order of minutes, and the time required for subsequent generation of a semantic network is not expected to be significant in comparison. We anticipate that generation of a semantic network representation for domain objects such as web pages should take roughly the same amount of time at worst. It is worth noting that the two Semantic Network Generators operate independently of each other, and could be run on separate processors.

It is also worth noting that a chart parser has polynomial time complexity [Jurafsky and Martin, 2000], while graph matching requires time that increases exponentially [Johnsonbaugh, 1984]. Because our architecture consists of a chart parser providing input to a graph matcher, it might appear that the time complexity of our

architecture would be exponential. However, computational complexity is a description of the worst possible case, and as mentioned above, graph matching can be done in linear time for "most" graphs. It is also important to note that natural language sentences of 30 words or more are uncommon [Covington, 1994] so that we are very close to the origin on any curve describing complexity. The result is that for our application, the difference between an exponential and a polynomial time algorithm is not expected to be significant from a practical point of view.

In contrast, generating and making effective use of formal representations for natural-language will certainly be a challenging task. A significant problem is that there are usually many ways to express a given statement of policy; that is, a given concept can often be expressed using many synonymous words, but also a variety of sentence structures. For our architecture to operate effectively, we require that all of these "synonymous statements" be transformed into the same formal semantic network representation. A lesser, but still significant concern is that different statements of similar wording or structure be converted into distinct and appropriate representations.

B. PRELIMINARY WORK: ERROR TYPES AND CAUSES

In a broad sense, there are only two types of errors that our architecture might commit: false negatives, and false positives. A false negative is said to have occurred when a *no* is returned by the Graph Matcher when the proper answer is *yes*. A false positive occurs when a *yes* is returned when the proper answer is *no*. In each case, the *yes* or *no* refers to whether or not there is a match between the semantic network representations of a policy statement and a domain object. These errors will occur when the semantic network representation for a policy statement and/or domain object does not accurately reflect its true characteristics.

A false negative occurs when the two graphs should match, but for some reason do not. As a consequence of a false negative, it would appear that a domain object (such as a web site) violated a policy statement, when in fact it did not. A false negative may occur for a number of reasons, including the following.

- If the semantic network representation for either a policy statement or a domain object is incomplete or is otherwise missing important information, it is likely, but not certain, that an error will occur. In the case where the graphs should not match, and the omission does not cause them to match, the report from the graph matcher will happen to be correct. However, a missing node or link from one of the two semantic networks will prevent a match from occurring so that a domain object that does conform to policy will appear as if it does not.
- If the two semantic networks use different terminology to represent the same concept (that is, if the terminology in the two semantic networks is inconsistent), then it may incorrectly appear that policy is being violated.

Conversely, a false positive occurs when the graphs should not match, but for some reason do. As a consequence of a false positive, a domain object (such as a web site) that conformed to a policy statement would be classified as being in violation. A false positive may occur for a number of reasons, including the following.

- As suggested above, if the policy semantic network is incomplete such that it is missing a part that would have caused a violation (i.e., it is missing a part that would not 'fit' into the domain object semantic network), then an incorrect match will be indicated.
- Similarly, if the terminology used to create the two semantic networks is inconsistent such that nodes or links that are in fact different appear to be the same, then it may incorrectly appear that the two graphs can be matched.

In considering whether false positives or false negatives lead to the more serious consequences, we might reasonably assume that for the most part, domain objects conform to policy, and that our architecture provides the most significant information when it identifies violations of policy. Under this assumption, false positives would be the most costly type of error, because actual violations would go undetected. That is, among the many true positives indicating the many domain objects conforming to a particular policy statement, there would be distributed a small number of false positives, each corresponding to a domain object which is in violation of policy, but which is

reported as being in conformance. Conversely, under this assumption false negatives would be relatively (though not completely) harmless: the suspect domain object would be investigated and acquitted from being in violation, with the only resultant cost being the time spent in the investigation. Under the (somewhat odd) assumption that domain objects only rarely conformed to policy, we reach the symmetrically opposite conclusion that false negatives are the most costly and false positives are relatively harmless. In the case where domain objects show no preference for either conformance to, or violation of policy, both types of error would appear to be about equally costly.

As an aside, we would expect that a given system for detecting policy violations could "trade-off" false positives and false negatives. That is, for a given fixed system, the number of false positives could be reduced only through a corresponding increase in false negatives, with the total number of errors remaining about the same. Identifying how the parameters of a system must be changed to adjust this trade-off should be a part of any design.

There are a couple of potential approaches to reducing errors like these. One would be to generalize in some way what we mean by a "match" between two semantic networks. One of the simpler ways in which ambiguity can be reduced is through the use of a single representative term to stand for collections of synonyms.

Other types of ambiguity that can lead to errors may be more complex to deal with. Most statements of policy (as well as most statements in any language) can be expressed in a number of different ways using a number of different sentence structures. It is expected that different sentence structures expressing the same idea may produce meaning lists that are different. At the very least, we know that many natural-language statements contain extraneous elements, or terms that can be eliminated without changing the semantics of the sentence. For example, a policy statement might require "compliance with the provisions of Security Document XYZ." However, this dictate could more concisely require "compliance with Security Document XYZ," because the expression "the provisions of" is superfluous. Problems like this one can be fixed with a collection of rule-based transformations: after a meaning-list has been found, a collection of hand-constructed rules can be applied to eliminate known redundancies and to apply

simplifications. Standard synonyms can be substituted into a meaning list using the same technique.

C. PRELIMINARY WORK: A TAXONOMY FOR NATURAL-LANGUAGE POLICY STATEMENTS

Table 1 shows 13 statements of policy taken from a policy document, circa 2001, for web sites at the Naval Postgraduate School. Several of these policy statements are made up of more than one sentence, so that Table 1 contains a total of 16 sentences. (In subsequent discussions, we will distinguish between different sentences in a policy statement using a decimal notation; for example, the second sentence of Policy Statement 1 will be referred to as Policy Statement 1.2.) These statements form a small but representative cross section of that policy document and of statements that we would expect to find in policy documents covering the same domain. In this subsection we create a simple set of categories, or a taxonomy, within which these and other natural-language policy statements can be classified.

A careful reading of the statements in Table 1 reveals that many of them are not, strictly speaking, statements of policy. That is, many of these statements are not imperatives that place the reader or any other domain object under an obligation to carry out an action or satisfy a requirement. Furthermore, this is not due to the lack of supporting context; placing any of these "non-policy" statements back within their full original context would not change this aspect of them. However, most, if not all of the statements can be interpreted as policy statements that have been phrased in informal, non-standard ways; this is because many of these policy statements are essentially speech acts that are worded as suggestions, but are intended to be interpreted as statements of policy. It is reasonable to expect that many natural-language policy documents will contain phrases that can be properly interpreted only when viewed as speech acts.

Similarly, definitions commonly appear in policy documents. Though in a strict sense definitions are not statements of policy, they can make up an important part of a policy document, because proper interpretation of the surrounding policy statements requires correct use of the definitions. In fact, in some cases the dividing line between a

definition and a policy statement may not be clear. For example, after a description of a certain type of information we may be told that "Such information is designated FOUO." It could be argued that this a definition, but it could also be argued that this is a statement of policy. In any case, formal analysis of policy needs to take into consideration definitions contained within a policy document.

Statement 1	Top-level web pages should be considered a directory of information contained in subsequent pages and should not contain detailed subject matter. Specifics should reside on subsequent pages.
Statement 2	Avoid long lists of hot links on 'top-level' organizational pages; a single link to a separate links page is much more effective.
Statement 3	Top-level organizational pages should have the same "look and feel" so that users will be able to know when they have navigated off of the main pages.
Statement 4	Web pages are dynamic, evolving documents that can frequently change. "Under construction" notices should be used sparingly.
Statement 5	Designers should recognize that graphics consume significant bandwidth. Provide thumbnail graphics of large graphics with a link to the full version. Graphics will also load faster if the height and width are given in the IMG SRC tag.
Statement 6	Use numbered or bulleted lists to condense text and to break up the page visually.
Statement 7	If you choose to use a background image or background color, make sure your text is readable. White or light-colored backgrounds are the most readable.
Statement 8	Limit the number of different font styles and colors on a page. A good rule of thumb is to use no more than three different fonts on a page.
Statement 9	Text and graphics that move can be particularly annoying. Use blinking text, scrolling marquees, animated gifs and Java applets very sparingly, if at all.
Statement 10	Any web site collecting personal information must comply with the provisions of reference (d). Network identification and Internet protocol addresses are not considered personal data.

Table 1. Ten natural-language policy statements obtained from the policy set for web pages at the Naval Postgraduate School, circa 2001. Several of these statements consist of more than one sentence; there is a total of 18 sentences contained within the 10 statements.

As discussed above, many of these policy statements are couched as suggestions so that a proper understanding requires interpretation in terms of speech acts. Many of the statements are phrased as polite recommendations; as a consequence, the word "should" needs to be replaced by the word "must." Also, a suggestion that something is desirable must be interpreted as an imperative that must be accomplished. For example, Statement 7 in Table 1 makes some suggestions and observations regarding readable text; interpreted as a statement of policy, it essentially states that pages are readable to the extent that their color resembles white or black. To get a fuzzy measure of how much an arbitrary color differs from grey, and thus how "readable" it is, we have to find out how "far away from grey" that color is. One possible way to do this is to note that in the RGB color space, various shades of grey correspond to color vectors with equal red, green, and

blue components. Given an arbitrary color (a, b, c) in RGB space, our problem translates into finding the value of grey that is closest to it; that is, we want to find the grey vector (j, j, j) such that the scalar magnitude $(a-j)^2+(b-j)^2+(c-j)^2$ is a minimum.

We created a simple classification, or taxonomy, of natural-language policy statements as shown in Table 2. The taxonomy appears to be comprehensive, in the sense that all statements from Table 1 can be placed into (at least) one of the six categories. Only one category in Table 2 does not have a representative from Table 1: that category, Category 5 contains titles, section headings, captions for figures and tables, and other text that is a part of a policy document, but that is not intended to be interpreted as policy. These statements can be thought of as summaries that indicate the type of information to be found in certain specified areas, and they may provide important guidance that could be exploited by a natural-language processor.

Category No.	Category Description	Examples from Table 1
1	Statements of Fact or Opinion	Statements 5.1, 7.2
2	Statements of Policy, including Recommendations, Statements of Advice, and Other Speech Acts that can be Interpreted as Policy	Statements 1.1, 1.2, 2, 3, 4.1, 4.2, 5.2, 5.3, 7.1, 8.1, 8.2, 9.1 9.2
3	Definitions	Statements 7.2, 10.1
4	Statements of Meta-Policy	Statement 10.2
5	Formatting and Other Non-Policy Statements	Not represented in Table 1

Table 2. A simple breakdown, or taxonomy, of the sentences that appear in the policy statements of Table 1. All sentences from Table 1 fall into one of the first four categories. Note that this table uses our convention in which decimal notation indicates the different sentences of a single policy statement; for example, Statement 5.2 refers to the second sentence of Policy Statement 5, etc.

Some of the items in Table 1 are simple observations of fact, or simple opinions. Such statements may imply policies that are very permissive of exceptions. However, for opinions or observations of fact, it may be less clear exactly what the intended statement of policy is. A study of Table 1 indicates that converting a statement of fact or opinion into a policy requires some nontrivial "creative thought" on the part of the converter.

Our taxonomy appears to be comprehensive, but it does not have particularly strong discriminating power: that is, we expect to see a lot of overlap between categories. For example, sentence 7.1, "If you choose to use a background image or

background color, make sure your text is readable," is classified as a statement of advice because of the phrase "make sure"- the very presence of this phrase in a policy statement implies that some leeway is permissible. However a consideration of speech acts makes it clear that the proper interpretation of this sentence is as a policy.

The statements in Table 1 suggest that some ambiguities will be relatively easy to deal with: in statement 7, "Provide thumbnail graphics of large graphics with a link to the full version," we might define the fuzzy term "large" to mean "greater than 250 kb." We may even get more refined by measuring the sizes of all graphics on a web site, and classifying as 'large' any graphic with size that falls in the top 30%. Operators for manipulating fuzzy terms, as discussed in Section II.G.2 could then be applied.

Because fuzzy descriptors are so valuable in dealing with ambiguities in natural-language text, we identified and characterized all the fuzzy descriptors that appear in our collection of policy statements. As shown in Table 3, we characterized each fuzzy descriptor through its grammatical category, the dimension along which it is fuzzy, and the spot or interval along the fuzzy dimension that the fuzzy descriptor specifies.

Fuzzy Descriptor	Grammatical Category	Fuzzy Dimension	Where on Dimension the Fuzzy Descriptor Applies
detailed (Statement 1)	Adjective	Amount of specific information	Large amounts of specific info
specifics (Statement 1)	Noun	Size of parts that make up whole	Small sized parts
long (Statement 2)	Adjective	Number of links	Many links
hot (Statement 2)	Adjective	Popularity of a web link	Very popular
effective (Statement 2)	Adjective	Capability for producing a result	Very capable
look and feel (Statement 3)	Noun	Appearance and interface mechanisms vs. Content	Emphasis on appearance and interface mechanisms
main (Statement 3)	Adjective	Place in time sequence of pages visited by a typical user	First or close to first
dynamic (Statement 4)	Adjective	Amount of movement or change	Large amount of movement
evolving (Statement 4)	Adjective	Constant vs. Intermittant change	Constant change
frequently (Statement 4)	Adjective	Number of changes per unit time	Large number of changes
under construction (Statement 4)	Adjective	Amount of work remaining to be done	Large amount of work remaining to be done
significant (Statement 5)	Adjective	Amount	Large amount
thumbnail (Statement 5)	Adjective	Physical size of a graphic	Small physical size
large (Statement 5)	Adjective	Size of a graphic file	Large file size
full (Statement 5)	Adjective	Fraction of a whole	At or near 100%
faster (Statement 5)	Adverb	Amount of time required in comparison to average case	Less time required
condense (Statement 6)	Verb	Degree to which something is made smaller	Make smaller to a moderate degree
break up (Statement 6)	Verb	Number of parts into which a whole is decomposed	A small number (approximately two to six)
background (Statement 7)	Adjective	Containing information vs. Not containing information	Containing little or no information
readable (Statement 7)	Adjective	Easy to read vs. Not easy to read	Easy to read
white (Statement 7)	Adjective	Strong color vs. Free from color	Free from color
light-colored (Statement 7)	Adjective	Strong color vs. Free from color	Almost free from color
most (Statement 7)	Adjective	Degree to which a characteristic (such as readability) holds	Characteristic holds strongly
limit (Statement 8)	Verb	Amount that is to be allowed	Amount allowed is little or none
particularly (Statement 9)	Adjective	Amount	More than almost all others
annoying (Statement 9)	Adjective	Pleasant vs. Unpleasant	Unpleasant
very (Statement 9)	Adverb	Emphasize following term vs. De-emphasize following term	Emphasize
sparingly (Statement 9)	Adverb	Frequency of appearance	Low frequency
if at all (Statement 9)	Adverbial Phrase	Degree to which something may be used	To be used very little, or not to be used
personal (Statement 10)	Adjective	Relating to a person as an individual vs. Relating to a group that a person belongs to	Closely related to a person as an individual

Table 3. Fuzzy descriptors and their characteristics from the policy statements of Table 1.

The entries in Table 3 provide valuable information about how values for fuzzy descriptors can be established for a given web site or page. In general, the value of a fuzzy descriptor can be established for a particular web site or page by taking

measurements over a population of such entities, and then by finding where the particular web site or page falls in the resulting distribution. For example, in Policy Statement 1, we can quantify the amount of "detailed subject matter" on a top-level web page by measuring, over a representative population of web sites, the amount of text on the top-level web pages. A top-level page that had more text than, say, 80% of the population would have a fuzzy value of 0.8 for "detailed subject matter." To determine whether or not a fuzzy descriptor applies, a threshold could be specified; for example, a top-level page containing more text than 75% of the pages in the population could be said to contain "detailed subject matter." Though the amount of text on a web page provides only a coarse measure of the amount of information it contains, it has the significant practical advantage that it is relatively easy to measure.

Most of the fuzzy descriptors in Table 3 can be established in a like manner. Whether or not a list is "long", a link is "hot," or a graphic is "large" can be established by the position of each entity within the population. Vector-space methods can be used to some extent in dealing with colors and color-spaces. Whether a certain type of notice is "used sparingly" could be determined by counting the fraction of pages on a site where the notice appears; if the fraction is less than some threshold, say 0.2, it could be said that the notice has in fact been used sparingly. Determining whether two web pages have the same "look and feel" might be accomplished through the use of a list of the characteristics that make up the "look and feel" of a web page. A count of places where the lists of two web pages differ could provide a coarse measure of whether they share the same "look and feel." Weightings could be assigned to characteristics on the list to provide a more refined measure of difference.

D. CASE STUDY: ANALYSIS OF NATURAL-LANGUAGE POLICY STATEMENTS

The previous subsections have contained preliminary discussions of our proposed representation and proof technique for formal analysis of natural-language policy statements. Though these discussions have helped us establish a vocabulary with which to work, and have even provided some insights into the potential strengths and

weaknesses of our approach, the following two important questions merit further consideration.

- What sort of ambiguities can we expect to see in natural-language policy documents? How should these ambiguities be dealt with?
- Will it be possible to extract from the policy statement the domain to which the policy applies? If not, how will this issue be dealt with?

To answer these questions, a case study was carried out on the ten natural-language policy statements in Table 1. In preparation for this case study, during the winter of 2002 Professor N. Rowe generated meaning lists for the 18 sentences contained in Table 1. This was accomplished by passing those statements through the MARIE parser [Guglielmo and Rowe, 1996]. The MARIE parser had previously been updated with new parse rules and dictionary entries. These updates to the MARIE parser were made before the statements contained in Table 1 had been obtained; thus the meaning lists presented below are representative of what the MARIE parser would generate from arbitrary natural language policy statements.

Our case study is presented in the remainder of this subsection. For each of the thirteen natural-language policy statements, we have determined the following particulars.

a) The Statement and its Meaning List. The meaning list was placed in a table in text format, and for some meaning lists a graphical representation was generated. Also, we noted whether the statement was one of operational, goal-oriented, or meta-policy, as was the statement's place in our taxonomy.

b) Authors Intention. In some cases, the intention of the policy statements author was unclear. We made an attempt to interpret the statement in a "reasonable" way.

c) Entities, Relationships, and Attributes. Entities, attributes, and relationships are taken from the meaning lists. *Entities* are anything that appears as the second

argument to the binary predicate "a_kind_of." *Relationships* are the predicate names, and *attributes* are the second arguments to property predicates.

d) Domain of Policy Statement. The domain of the policy statement was established; that is, we determined the domain objects that would have to be put into semantic network form for conformance to this policy statement to be checked.

e) Modal Auxiliaries. Modal auxiliaries were identified.

f) Fuzzy Descriptors. Quantifiers, whether or not fuzzy, were identified.

g) Anaphoric References. Anaphoric references were identified and resolved.

h) Speech Act Theory. Speech acts contained within the policy statement were identified and described.

i) Miscellaneous Observations. In some cases, there are significant observations that fit nowhere else are discussed here here. In cases where there are no such observations, this heading is dropped.

Words appearing in meaning lists are encoded with Wordnet sense numbers. The encoding is based on the sum of two numbers: one number indicating the part of speech, and the other number indicating the particular word sense from Wordnet. The numbers indicating parts of speech consist of zero for nouns, 50 for adjectives, 100 for verbs, 150 for adverbs, and 199 for other parts of speech. The encoding can be clarified through an example. The term "see-108" indicates a meaning which is the same as that of the eighth verb sense of the word "see" in Wordnet. Predicates in meaning lists are assertions that the first argument has the property indicated by the head of the predicate, with the value that appears as the second argument. In an informal sense, the head of a predicate can be placed between the two arguments to make a valid assertion; for example, the predicate "a_kind_of(v4,substance-2)" can be informally interpreted to mean that "constant v4 is a kind of substance." Terms of the form "vXXX," where X is a digit, are constants. These parser-invented constant names are used to represent distinct entities. Variable names have significance only within individual meaning lists.

It should be noted that in all graphical representations of meaning lists, the arrows point from the second term in a binary relationship to the first, and from an attribute value to the entity that holds the attribute (just as in Figures 8 and 9). Appendix A contains documentation for the software used to generate the graphical representations of meaning lists.

1. Policy Statement 1

a) The Statement and its Meaning List. Policy Statement 1 is: "Top-level web pages should be considered a directory of information contained in subsequent pates and should not contain detailed subject matter. Specifics should reside on subsequent pages." Both the first and second sentences of this policy statement fit into our taxonomy as recommendations or statements of advice. The meaning lists for this policy statement are given in text form in Tables 4 and 5, and in graphical format in Figures 10 and 11.

First Sentence of Policy Statement 1	Top-level web pages should be considered a directory of information contained in subsequent pages and should not contain detailed subject matter.
Meaning List	[a_kind_of(v886,see-108),tense(v886,past), marker(v886,passive), tense(v886,should), object(v886,v898), a_kind_of(v898,directory-1), quantification(v898,a), part_of(v898,v903), a_kind_of(v903,information-2), object(v912,v903), a_kind_of(v912,contain-103), tense(v912,pastpart), inside(v912,v936), a_kind_of(v936,page-1), quantification(v936,plural), property(v936,subsequent-51), a_kind_of(v944,contain-103), quantification(v944,plural), property(v944,not-151), tense(v944,should), object(v944,v4), a_kind_of(v4,substance-2), property(v4,detailed-52), agent(v886,v874), agent(v944,v874), a_kind_of(v874,page-1), quantification(v874,plural), part_of(v874,v865), a_kind_of(v865,web-0), has_property(v874,v342), a_kind_of(v342,level-1), property(v342,top-52), distinct(v886,v944)]

Table 4. The meaning list for the sentence "Top-level web pages should be considered a directory of information contained in subsequent pages and should not contain detailed subject matter."

Second Sentence of Policy Statement 1	Specifics should reside on subsequent pages.
Meaning List	[a_kind_of(v8,live-101), quantification(v8,plural), tense(v8,should), on(v8,v156), a_kind_of(v156,page-1), quantification(v156,plural), property(v156,subsequent-51), agent(v8,v1), a_kind_of(v1,specific-0), quantification(v1,plural)]

Table 5. The meaning list for the second sentence of Policy Statement 1: "Specifics should reside on subsequent pages."

b) Authors Intention. The authors intention here is clear: this policy statement requires that web sites have a structure such that a page at or near the entry point or "home page" of the site provides organized access to other pages in the web site.

c) Entities, Relationships, and Attributes for First Sentence of Policy Statement 1:

Entities: a_kind_of(v886,see-108)
a_kind_of(v898,directory-1)
a_kind_of(v903,information-2)
a_kind_of(v912,contain-103)
a_kind_of(v936,page-1)
a_kind_of(v944,contain-103)
a_kind_of(v4,substance-2)
a_kind_of(v874,page-1)
a_kind_of(v865,web-0)
a_kind_of(v342,level-1)

Relationships: a_kind_of,tense, marker, object, quantification, part_of, inside, property, agent,
has_property, distinct

Attributes:
tense(v886,past),
marker(v886,passive),
tense(v886,should),
object(v886,v898),
quantification(v898,a),
tense(v912,pastpart),
quantification(v936,plural),
property(v936,subsequent-51),
quantification(v944,plural),
property(v944,not-151),
tense(v944,should),
property(v4,detailed-52),
quantification(v874,plural),
part_of(v874,v865),

property(v342,top-52)

Entities, Relationships, and Attributes for Second Sentence of Policy Statement 1:

Entities: a_kind_of(v8,live-101),
a_kind_of(v156,page-1),
a_kind_of(v1,specific-0)

Relationships: a_kind_of, quantification, tense, on, property, agent

Attributes:

quantification(v8,plural),
tense(v8,should),
quantification(v156,plural),
property(v156,subsequent-51),
quantification(v1,plural)

d) Domain of Policy Statement. This policy statement has a domain consisting of entire web sites.

e) Modal Auxiliaries. The modal auxiliary should appears three times in this policy statement: twice in the first sentence, and once in the second sentence.

f) Fuzzy Descriptors. "Detailed" and "specifics" are fuzzy descriptors.

g) Anaphoric References. The term "specifics" in the second sentence refers to the "detailed subject matter" mentioned in the previous sentence.

h) Speech Act Theory. The word "should," is used in this policy statement as a polite way of stating what must hold, and the expression "...should be considered..." is a polite way of saying that web designers are required to ensure that top-level web pages have the characteristics of a directory of information. More importantly, this policy statement implies that it is common for web designers to put too much detail in their top-level pages. In most, if not all of the policy statements that we'll examine, statements made by the policy author address misunderstandings that web designers are perceived to have.

i) Miscellaneous Comments. The statement of policy "specifics must reside on subsequent pages" can be misinterpreted with improper usage of the term "must." What is really being said here is that "specifics, if they exist, must reside on subsequent pages;" there is no need to generate specifics to place on subsequent pages if they do not already

exist. Perhaps a clearer way to phrase the statement is that "specifics may not reside on the page in question." Assuming that we can determine exactly what constitutes "specifics," probably the most effective way to check whether this policy is being conformed to is to create a semantic network representation for "specifics," and to see if the pages specified by the policy statement contain them or otherwise have them as attributes; if so, the policy is being violated.

A reasonable simplification of the first sentence would eliminate the phrase "should be considered a directory of information contained in subsequent pages," and would instead simply state that "top-level web pages must not contain detailed information." Also, we interpret the word "subsequent" to mean "subsequent in time." That is, a "subsequent" page is one that a typical user would navigate to after the top-level page. We can infer that a "subsequent page" is one that is connected to the top-level page by some link sequence.

Finally, the phrase "detailed subject matter" in the first sentence can be simplified into "details." This transformation can be represented with the following rule.

detailed subject matter ⇒ details

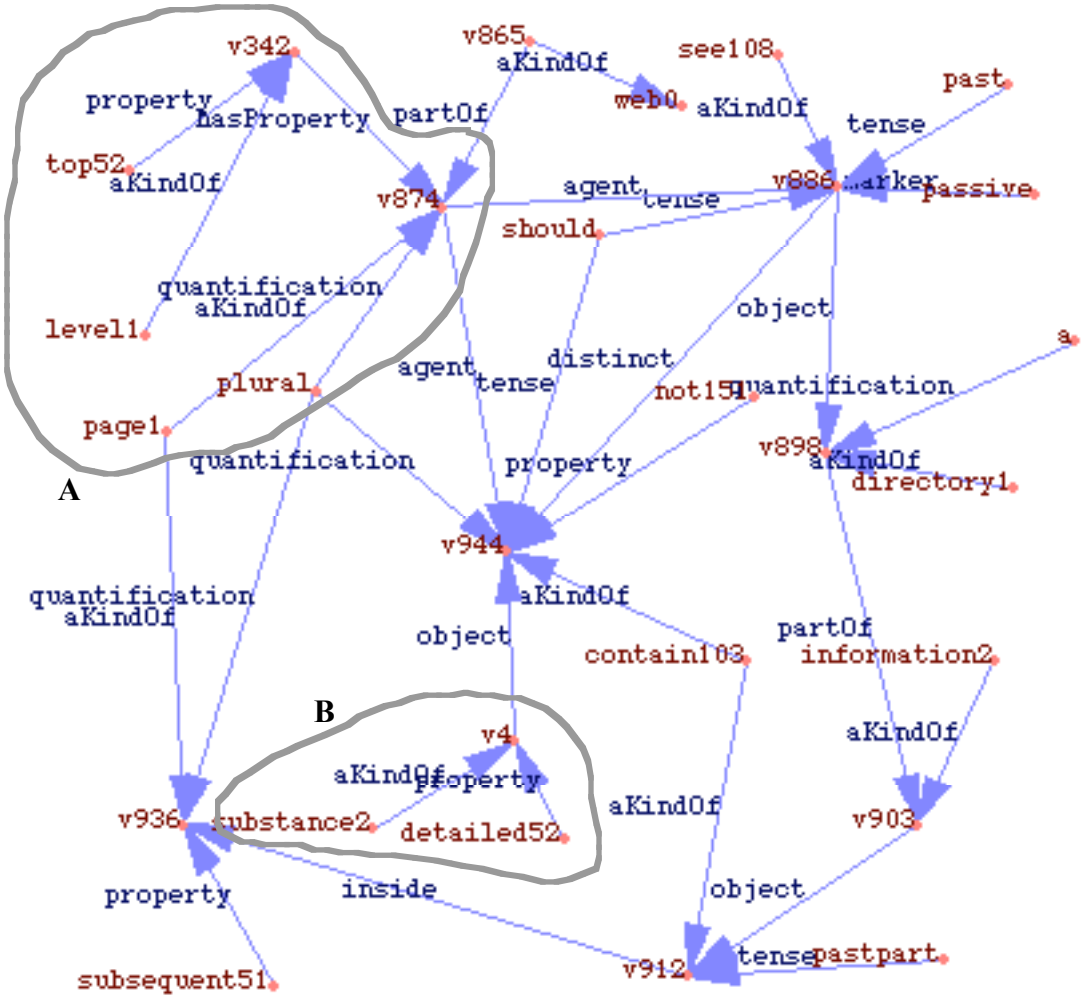

Figure 10. Graphical representation of the meaning list for Policy Statement 1.1: "Top-level web pages should be considered a directory of information contained in subsequent pages and should not contain detailed subject matter." A semantic network representation for "Top-level web pages" is enclosed in the grey loop labeller "A," and a semantic network representation for "detailed subject matter is enclosed in the grey loop labeller "B."

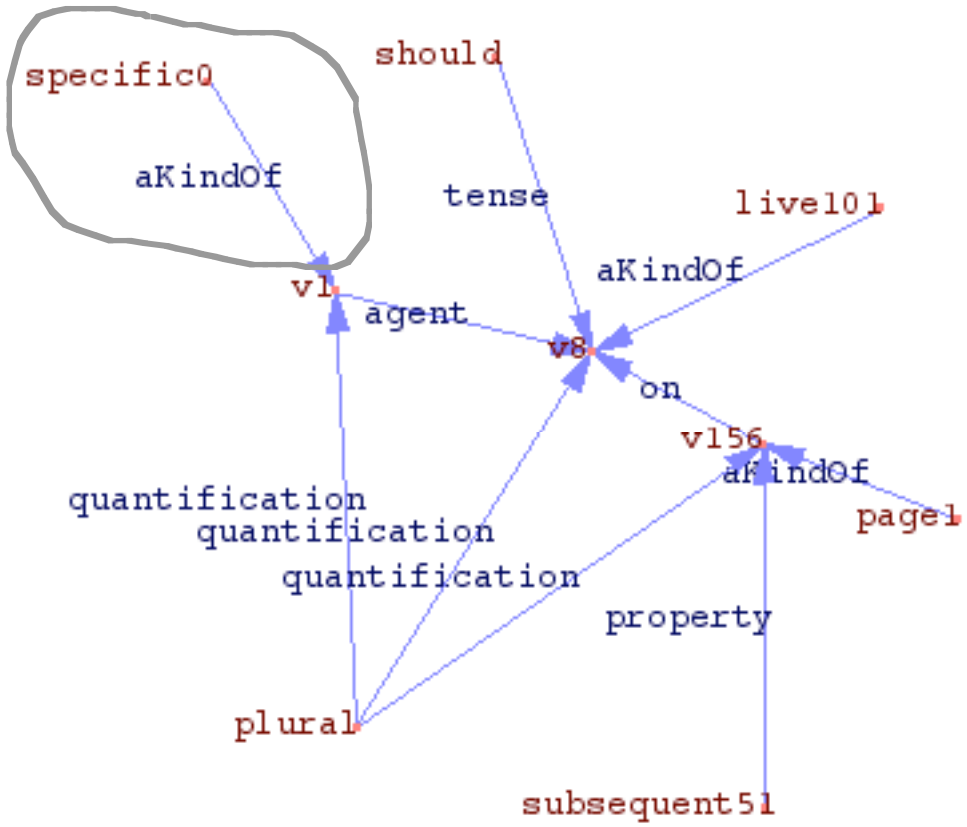

Figure 11. Graphical representation of the meaning list for Policy Statement 1.2: "Specifics should reside on subsequent pages."

2. Policy Statement 2

a) The Statement and its Meaning List. Policy Statement 2 is "Avoid long lists of hot links on 'top-level' organizational pages; a single link to a separate links page is much more effective." Policy Statement 2 fits into our taxonomy as a recommendation or statement of advice that can be interpreted as a policy statement. The meaning list for this policy statement is given in text form in Table 6.

Policy Statement 2	Avoid long lists of hot links on 'top-level' organizational pages; a single link to a separate links page is much more effective.
Meaning List	[a_kind_of(v4,avoid-101), quantification(v4,plural), object(v4,v31), a_kind_of(v31,register-1), quantification(v31,plural), property(v31,long-52), subject(v31,v78), a_kind_of(v78,link-0), quantification(v78,plural), property(v78,hot-55), on(v78,v601), a_kind_of(v601,page-1),quantification(v601,plural), property(v601,organizational-51), has_property(v601,v448), a_kind_of(v448,level-1),property(v448,top-52), a_kind_of(v726,be-101),property(v726,effective-54), relationship(effective-54,v753), property(v753,much-151), property(v753,more-151), agent(v726,v633), a_kind_of(v633,link-0), property(v633,single-51), quantification(v633,a), to(v633,v716), a_kind_of(v716,page-1), agent(v703,v716), a_kind_of(v703,link-0), quantification(v703,plural), property(v703,separate-51), quantification(v716,a)]

Table 6. The meaning list for Policy Statement 2: "Avoid long lists of hot links on 'top-level' organizational pages; a single link to a separate links page is much more effective."

b) Authors Intention. The authors intention here is clear: a separate "links" page is required, and mixing links with other types of information is forbidden, especially on pages at or near the home page.

c) Entities, Relationships, and Attributes for Policy Statement 2:

Entities:

a_kind_of(v4,avoid-101),
a_kind_of(v31,register-1),
a_kind_of(v78,link-0),
a_kind_of(v601,page-1),
a_kind_of(v448,level-1),
a_kind_of(v726,be-101),
a_kind_of(v633,link-0),
a_kind_of(v716,page-1),
a_kind_of(v703,link-0),

Relationships: a_kind_of, quantification, object, property, subject, on, has_property, relationship, agent, to

Attributes:

quantification(v4,plural),
quantification(v31,plural),
property(v31,long-52),
quantification(v78,plural),
property(v78,hot-55),

quantification(v601,plural),
property(v601,organizational-51),
property(v448,top-52),
property(v726,effective-54),
relationship(effective-54,v753),
property(v753,much-151),
property(v753,more-151),
property(v633,single-51),
quantification(v633,a),
quantification(v703,plural),
property(v703,separate-51),
quantification(v716,a)

d) Domain of Policy Statement. This policy statement applies to web sites as a whole.

e) Modal Auxiliaries. None.

f) Fuzzy Descriptors. Fuzzy descriptors are "long," "hot," and "effective."

g) Anaphoric References. "Top-level organization pages" is an anaphoric reference to the top-level pages in Policy Statement 1.

h) Speech Act Theory. Saying that "X should be avoided" is being used here to say that X should not occur. Also, the statement that "Y is more effective" is a polite way of saying that Y is required.

i) Miscellaneous Comments. An ellipsis phenomenon (the omission or suppression of parts of a word or sentence) occurs in this policy statement: "a single link to a separate links page" gives a destination, but the source of the link is not explicitly stated, and the reader has to infer that the source is the "top-level page."

Also, the following simplifying transformation should be applied.

X is more effective ⇒ X should be done

3. Policy Statement 3

a) The Statement and its Meaning List. Policy Statement 3 is "Top-level organizational pages should have the same 'look and feel' so that users will be able to know when they have navigated off of the main pages." It fits into our taxonomy as a

recommendation or statement of advice that can be interpreted as a statement of policy. The meaning list for this policy statement is given in text form in Table 7.

Policy Statement 3	Top-level organizational pages should have the same 'look and feel' so that users will be able to know when they have navigated off of the main pages.
Meaning List	[a_kind_of(v903,have-101), quantification(v903,plural), tense(v903,should), object(v903,v4), a_kind_of(v4,'look and feel'-0), property(v4,quasi-51),property(v4,same-51), quantification(v4,the), so(v903,v7), a_kind_of(v7,be-101), tense(v7,future), object(v7,v1164), a_kind_of(v1164,entity-1), property(v1164,able-51), for(v7,v1184), a_kind_of(v1184,know-103), object(v1184,v2527), a_kind_of(v2527,period-2), during(v1194,v2527), a_kind_of(v1194,navigate-103), tense(v1194,past), tense(v1194,perfect), off(v1194,v1255), a_kind_of(v1255,page-1), property(v1255,primary-56), quantification(v1255,the), agent(v1194,v1199), a_kind_of(v1199,people-1), anaphoric(v1199), tense(v1184,infinitive), agent(v7,v948), a_kind_of(v948,user-1), quantification(v948,plural), agent(v903,v894), a_kind_of(v894,page-1), quantification(v894,plural), property(v894,organizational-51), has_property(v894,v377), a_kind_of(v377,level-1), property(v377,primary-56)]

Table 7. The meaning list for the sentence "Top-level organizational pages should have the same 'look and feel' so that users will be able to know when they have navigated off of the main pages."

b) Authors Intention. This policy statement requires that pages on a web site, and especially at or near the home page, have the same "look and feel." Except for the definition of exactly what qualifies as "look and feel," the authors intention here is clear.

c) Entities, Relationships, and Attributes for Policy Statement 3:

Entities:

a_kind_of(v903,have-101),
a_kind_of(v4,'look and feel'-0),
a_kind_of(v7,be-101),
a_kind_of(v1164,entity-1),
a_kind_of(v1184,know-103),
a_kind_of(v2527,period-2),
a_kind_of(v1194,navigate-103),
a_kind_of(v1255,page-1),
a_kind_of(v1199,people-1),
a_kind_of(v948,user-1),
a_kind_of(v894,page-1),

a_kind_of(v377,level-1)

Relationships: a_kind_of, quantification, tense,object, property, so,or, during, off, agent, anaphoric,

has_property

Attributes:

quantification(v903,plural),
tense(v903,should),
property(v4,quasi-51),
property(v4,same-51),
quantification(v4,the),
tense(v7,future),
property(v1164,able-51),
tense(v1194,past),
tense(v1194,perfect),
property(v1255,primary-56),
quantification(v1255,the),
tense(v1184,infinitive),
quantification(v948,plural),
quantification(v894,plural),
property(v894,organizational-51),
property(v377,primary-56)

d) Domain of Policy Statement. This policy statement applies to all web sites.

e) Modal Auxiliaries. The modal auxiliary *should* appears once in this policy statement. Also, "will be able to" should be interpreted as "can," and thus is a modal auxiliary.

f) Fuzzy Descriptors. "Look and feel" and "main" are fuzzy descriptors.

g) Anaphoric References. "Top-level organizational pages" is an anaphoric reference to Policy Statement 1.

h) Speech Act Theory. The author justifies this policy statement with the phrase "so that users will be able to know when they have navigated off of the main pages." Justification is not necessary in a statement of policy, and it has an interesting effect here: it moves this policy statement from being a purely operational statement of policy to one that is to some extent goal-oriented.

i) Miscellaneous Comments. The following transformations can be applied.

will be able to □ can
navigate off of □ leave

4. Policy Statement 4

a) The Statement and its Meaning List. Policy Statement 4 is "Web pages are dynamic, evolving documents that can frequently change. 'Under construction' notices should be used sparingly." The first sentence of this policy statement fits into our taxonomy as a statement of fact, and the second sentence fits into our taxonomy as a statement of advice. The meaning lists for this policy statement are given in text form in Tables 8 and 9. The meaning list for the first sentence of this policy statement is given in graphical form in Figure 12.

First Sentence of Policy Statement 4	Web pages are dynamic, evolving documents that can frequently change.
Meaning List	[a_kind_of(v30,be-108), quantification(v30,plural), object(v30,v55), a_kind_of(v55,document-2), quantification(v55,plural), property(v55,dynamic-52), agent(v50,v55), a_kind_of(v50,evolve-104), distinct(v44,v50), agent(v63,v55), a_kind_of(v63,alter-103), quantification(v63,plural), property(v63,frequently-151), tense(v63,can), agent(v30,v16), a_kind_of(v16,page-1), quantification(v16,plural), part_of(v16,v1), a_kind_of(v1,web-0)]

Table 8. The meaning list for the first sentence of Policy Statement 4: "Web pages are dynamic, evolving documents that can frequently change."

Second Sentence of Policy Statement 4	"Under construction" notices should be used sparingly.
Meaning List	[a_kind_of(v99,apply-107), property(v99,sparingly-150), quantification(v41,plural),tense(v99,past), tense(v99,should),object(v99,v24), a_kind_of(v24,notice- 2),quantification(v24,plural),object(v24,v16), a_kind_of(v16,construction-6), property(v16,under-50),property(v16,quasi-51)]

Table 9. The meaning list for the second sentence of Policy Statement 4: "'Under construction' notices should be used sparingly."

b) Authors Intention. The first sentence of this policy statement is a simple statement of fact. It provides motivation for the second sentence, which except for the

somewhat ambiguous term “sparingly,” is a clear statement of policy. Except for the term “sparingly,” the authors intention here is clear.

c) Entities, Relationships, and Attributes for First Sentence of Policy Statement 4.

Entities: a_kind_of(v30,be-108),
a_kind_of(v55,document-2),
a_kind_of(v50,evolve-104),
a_kind_of(v63,alter-103),
a_kind_of(v16,page-1),
a_kind_of(v1,web-0)

Relationships: a_kind_of, quantification, object, property, agent, distinct, tense, part_of

Attributes:
quantification(v30,plural),
quantification(v55,plural),
property(v55,dynamic-52),
quantification(v63,plural),
property(v63,frequently-151),
tense(v63,can),
quantification(v16,plural)

Entities, Relationships, and Attributes for Second Sentence of Policy Statement 4.

Entities: a_kind_of(v99,apply-107),
a_kind_of(v24,notice-2),
a_kind_of(v16,construction-6)

Relationships: a_kind_of, property, quantification, tense, object

Attributes:
property(v99,sparingly-150),
quantification(v41,plural),
tense(v99,past),
tense(v99,should),
quantification(v24,plural),
property(v16,under-50),
property(v16,quasi-51)

d) Domain of Policy Statement. This policy statement can be applied to any web site.

e) Modal Auxiliaries. The modal auxiliary “can” appears in the first sentence, and the modal auxiliary “should” appears in the second sentence of this policy statement.

f) Fuzzy Descriptors. Fuzzy descriptors in this policy statement are “dynamic,” “evolving,” “frequently,” “under construction,” and “sparingly” are fuzzy descriptors.

g) Anaphoric References. None.

h) Speech Act Theory. The term "should" is used in the second sentence of this policy statement as a polite substitute for the word "must." More significantly, by simply stating the first sentence of this policy statement, the author infers that some people responsible for web pages are not aware that pages frequently change.

i) Miscellaneous Comments. A key ellipsis is the omission of "on [web] pages" from the end of the last sentence.

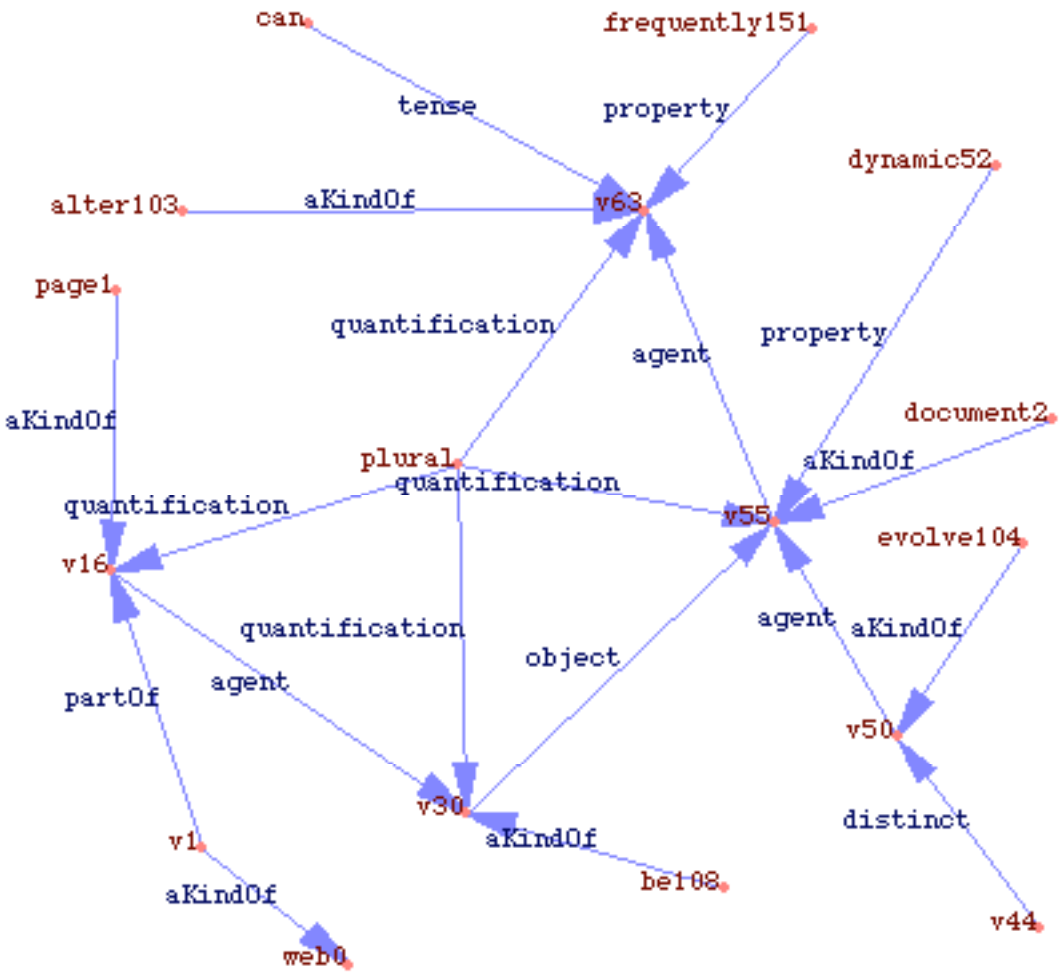

Figure 12. Graphical representation of the meaning list for the sentence "Web pages are dynamic, evolving documents that can frequently change."

5. Policy Statement 5

a) The Statement and its Meaning List. Policy Statement 5 is: "Designers should recognize that graphics consume significant bandwidth. Provide thumbnail graphics of

large graphics with a link to the full version. Graphics will also load faster if the height and width are given in the IMG SRC tag." The first sentence of this policy statement fits into our taxonomy as a statement of fact; the second sentence fits into our taxonomy as a statement of policy; and the third sentence fits into our taxonomy as a statement of fact. The meaning lists for this policy statement are given in text format in Tables 10, 11, and 12. The meaning lists for the first and second sentences of this statement are given in graphical form in Figures 13 and 14.

First Sentence of Policy Statement 5	Designers should recognize that graphics consume significant bandwidth.
Meaning List	[a_kind_of(v36,recognize-108), quantification(v36,plural), tense(v36,should), agent(v36,v1), a_kind_of(v1,architect-1), quantification(v1,plural), object(v36,v55), a_kind_of(v55,exhaust-102), quantification(v55,plural), object(v55,v76), a_kind_of(v76,bandwidth-1), property(v76,substantial-56), agent(v55,v60), a_kind_of(v60,graphics-2)]

Table 10. The meaning list for the first sentence of Policy Statement 5: "Designers should recognize that graphics consume significant bandwidth."

Second Sentence of Policy Statement 5	Provide thumbnail graphics of large graphics with a link to the full version.
Meaning List	[a_kind_of(v4,render-101), quantification(v4,plural), tense(v4,imperative), object(v4,v10), a_kind_of(v10,graphics-2), a_kind_of(v10,thumbnail-0), object(v10,v25), a_kind_of(v25,graphics-2), property(v25,big-51), beside(v10,v48), a_kind_of(v48,link-0), quantification(v48,a), range_to(v48,v91), a_kind_of(v91,version-2), property(v91,full-57), quantification(v91,the)]

Table 11. The meaning list for the second sentence of Policy Statement 5: "Provide thumbnail graphics of large graphics with a link to the full version."

Third Sentence of Policy Statement 5	Graphics will also load faster if the height and width are given in the IMG SRC tag.
Meaning List	[a_kind_of(v24,load-103),quantification(v24,plural), property(v24,faster-151),property(v24,also-151), tense(v24,future),agent(v24,v6), a_kind_of(v6,graphics-2),if(v46,v24), a_kind_of(v46,give-105), quantification(v68,plural),tense(v46,past), inside(v46,v3),a_kind_of(v3,'img src'-0), quantification(v3,the),object(v46,v54), object(v46,v63),a_kind_of(v54,height-3), a_kind_of(v63,width-1),distinct(v54,v63)]

Table 12. The meaning list for the third sentence of Policy Statement 5: "Graphics will also load faster if the height and width are given in the IMG SRC tag."

b) Authors Intention. This policy statement requires that large graphics have thumbnail graphics that stand in their stead, and that IMG SRC tags be used in the HTML constructs for graphics. The authors intention here is clear.

c) Entities, Relationships, and Attributes for First Sentence of Policy Statement 5.

Entities: a_kind_of(v36,recognize-108),
a_kind_of(v1,architect-1),
a_kind_of(v55,exhaust-102),
a_kind_of(v76,bandwidth-1),
a_kind_of(v60,graphics-2)

Relationships: a_kind_of, quantification, tense, agent, object, property

Attributes:
quantification(v36,plural),
tense(v36,should),
quantification(v1,plural),
quantification(v55,plural),
property(v76,substantial-56)

Entities, Relationships, and Attributes for Second Sentence of Policy Statement 5.

Entities: a_kind_of(v4,render-101),
a_kind_of(v10,graphics-2),
a_kind_of(v10,thumbnail-0),
a_kind_of(v25,graphics-2),
a_kind_of(v48,link-0),
a_kind_of(v91,version-2)

Relationships: a_kind_of, quantification, tense, object, property, beside, range_to

Attributes:
quantification(v4,plural),
tense(v4,imperative),
property(v25,big-51),
quantification(v48,a),

property(v91,full-57),
quantification(v91,the)

Entities, Relationships, and Attributes for Third Sentence of Policy Statement 5.

Entities: a_kind_of(v24,load-103),
a_kind_of(v6,graphics-2),
a_kind_of(v46,give-105),
a_kind_of(v3,'img src'-0),
a_kind_of(v54,height-3),
a_kind_of(v63,width-1)

Relationships: a_kind_of,quantification, property, tense,agent,if, inside, object, distinct

Attributes:
quantification(v24,plural),
property(v24,faster-151),
property(v24,also-151),
tense(v24,future),
quantification(v68,plural),
tense(v46,past),
quantification(v3,the)

d) Domain of Policy Statement. This statement applies to all pages on a web site. Information about the domain of this policy statement is not contained explicitly within the policy statement.

e) Modal Auxiliaries. The modal auxiliary "should" appears in the first sentence of this policy statement, and the modal auxiliary "will" appears in the second sentence.

f) Fuzzy Descriptors. Fuzzy descriptors are "significant," "thumbnail," "large," "full," and "faster."

g) Anaphoric References. The term "full version" in sentence two is an anaphoric reference to the term "large graphics."

h) Speech Act Theory. In the first sentence is not a policy statement, but a polite statement that web designers have a responsibility to not burden a user with gratuitous graphics. This sentence establishes that significant bandwidth is bad, and thus that graphics should be minimized. The word "will" in the third sentence implies that we should give height and width in the IMG SRC tag, because loading graphics faster is good.

i) Miscellaneous Observations. There is an ellipsis phenomena associated with the terms “height” and “width;” the full expressions are “height of the graphics” and “width of the graphics.” Also, the phrase “designers should recognize” should be eliminated, because it is implied that all of these policy statements consist of things that designers should recognize. This can be expressed with the following transformation rule.

designers should recognize $\rightarrow$ $\varnothing$

Thumbnail graphics appear within HTML in the following way [Castro, 2003]. The “image.jpg” is the path to the full sized image, and “mini.jpg” is the path to the thumbnail image. The “alt=...” term is an optional text phrase that will appear if the thumbnail image for some reason does not.

```
<a href= “image.jpg”>
<img src=“mini.jpg” alt=“text that appears if thumbnail does not” />
</a>
```

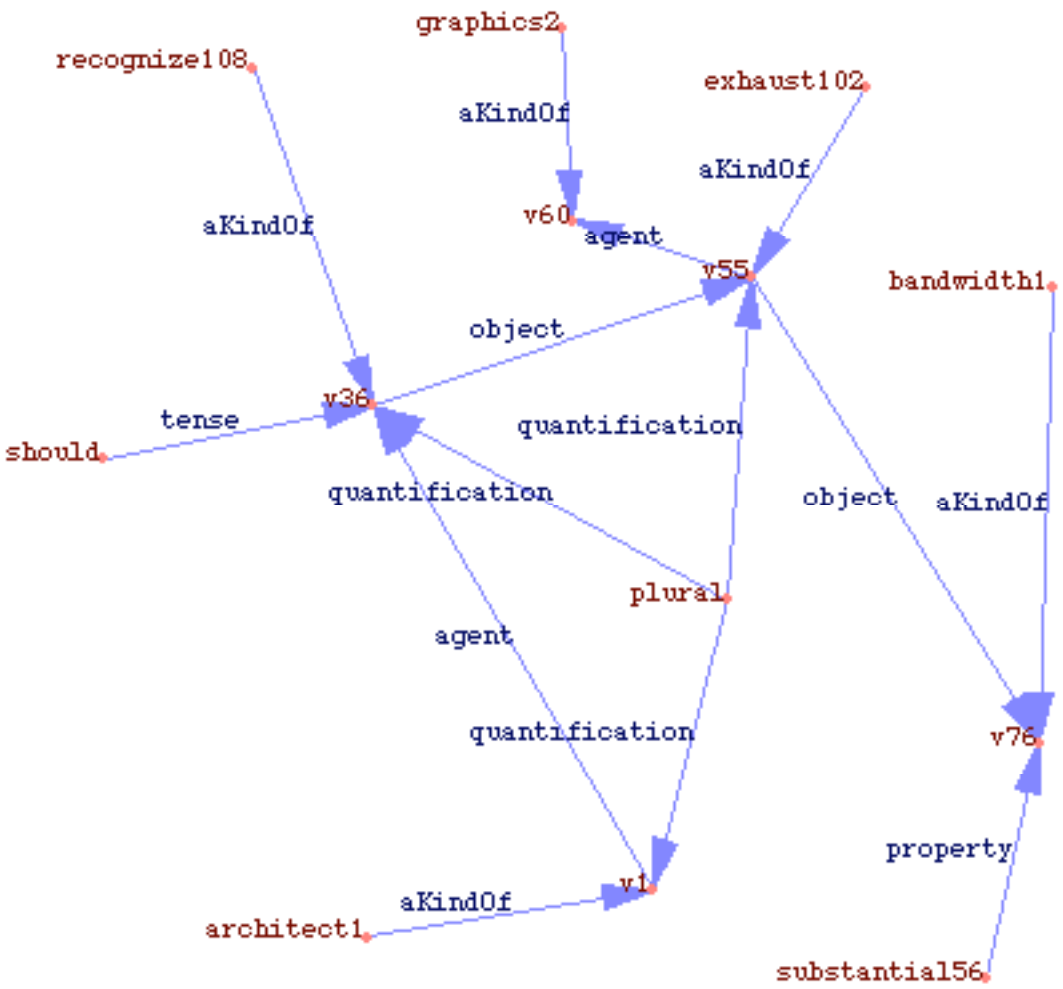

Figure 13. Graphical representation of the meaning list for the first sentence of Policy Statement 5.1: “Designers should recognize that graphics consume significant bandwidth.”

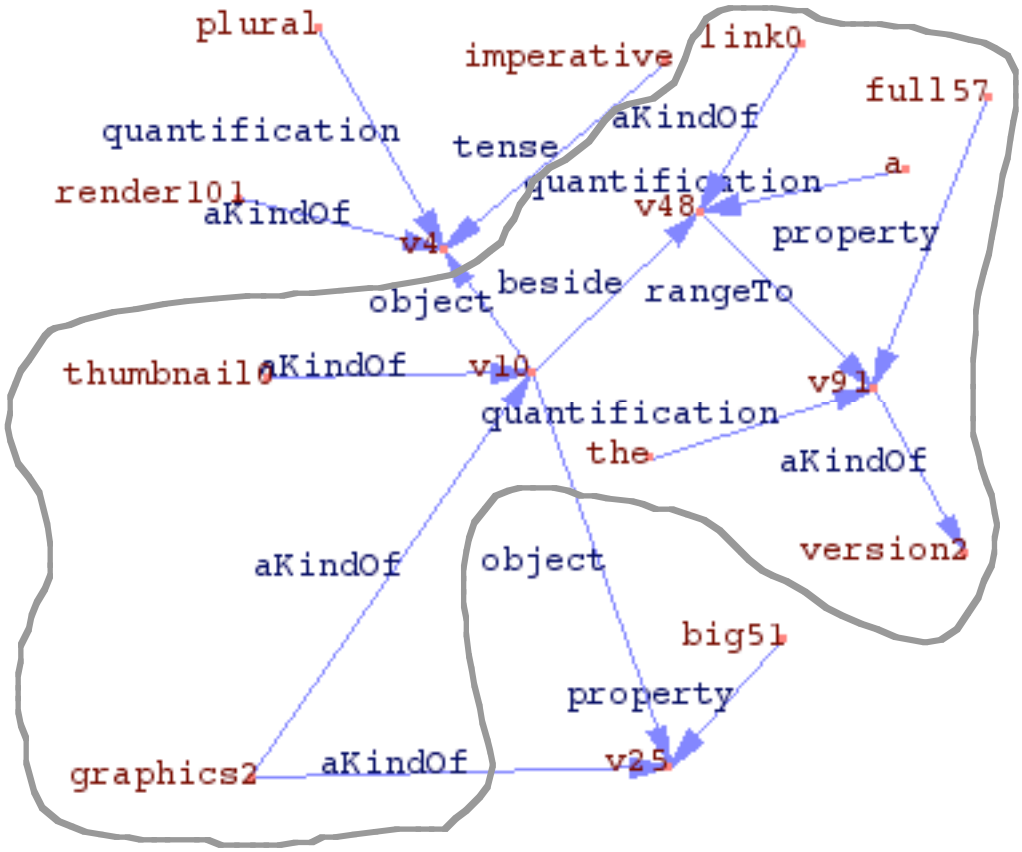

Figure 14. Graphical representation of meaning list for Policy Statement 5.2: "Provide thumbnail graphics of large graphics with a link to the full version." A semantic network representation for "thumbnail graphics with a link to the full version" is enclosed in the grey loop.

6. Policy Statement 6

a) The Statement and its Meaning List. Policy Statement 6 is: "Use numbered or bulleted lists to condense text and to break up the page visually." This sentence fits into our taxonomy as a simple statement of policy. The meaning list for this policy statement is given in text format in Table 13.

Policy Statement 6	Use numbered or bulleted lists to condense text and to break up the page visually.
Meaning List	[a_kind_of(v44,apply-107), quantification(v44,plural),tense(v44,imperative), object(v44,v68),a_kind_of(v68,list- 2),quantification(v68,plural),object(v5435,v68), conjunction(v5435,[v52,v61],or), tense(v5435,pastpart),a_kind_of(v52,number-102), a_kind_of(v61,bullet-100),for(v68,v82),for(v68,v89), a_kind_of(v82,condense-104), quantification(v82,plural), tense(v82,infinitive),object(v82,v85), a_kind_of(v85,text-2), a_kind_of(v89,'break up'-114), quantification(v89,plural), tense(v89,infinitive),object(v89,v182), a_kind_of(v182,page-1),quantification(v182,the), property(v89,visual-51),marker(v89,adverbial), distinct(v82,v89)]

Table 13. The meaning list for Policy Statement 6: "Use numbered or bulleted lists to condense text and to break up the page visually."

b) Authors Intention. Though this statement is an imperative to format text in a certain way, the circumstances under which this action should be carried out is not clearly specified in the statement itself. It is clearly not appropriate to use numbered or bulleted lists to the exclusion of any other type of text format (i.e., plain paragraphs are still allowed). Does a full page of text without any numbered or bulleted lists violate this policy? Probably so. But does a page of text that contains no numbered or bulleted lists, but which does contain several pictures that break up the page visually, violate this policy? Possibly not, especially if a bulleted or numbered list would clutter a page that is appealing as it stands.

c) Entities, Relationships, and Attributes for Policy Statement 6:

Entities: a_kind_of(v44,apply-107),
a_kind_of(v68,list-2),
a_kind_of(v52,number-102),
a_kind_of(v61,bullet-100),
a_kind_of(v82,condense-104),
a_kind_of(v85,text-2),
a_kind_of(v89,'break up'-114),
a_kind_of(v182,page-1)

Relationships: a_kind_of, quantification,tense, object, conjunction, for, object, property, marker, distinct

Attributes:

quantification(v44,plural),
tense(v44,imperative),
quantification(v68,plural),
tense(v5435,pastpart),
quantification(v82,plural),
tense(v82,infinitive),
quantification(v89,plural),
tense(v89,infinitive),
quantification(v182,the),
property(v89,visual-51)

d) Domain of Policy Statement. This policy applies to web pages.

e) Modal Auxiliaries. None.

f) Fuzzy Descriptors. The terms "condense" and "break up" are fuzzy descriptors.

g) Anaphoric References. None.

h) Speech Act Theory. This policy statement implies that condensing text and breaking up a page visually are good things to do most of the time.

i) Miscellaneous Comments. An ellipsis phenomena is evident in that numbered bulleted lists are on a web page.

7. Policy Statement 7

a) The Statement and its Meaning List. Policy Statement 7 is: "If you choose to use a background image or background color, make sure your text is readable. White or light-colored backgrounds are the most readable." The first sentence of this policy statement fits into our taxonomy as a statement of advice that can be converted into policy. The second sentence is a simple observation of fact. The meaning lists for this policy statement are given in text format in Tables 14 and 15.

First Sentence of Policy Statement 7	If you choose to use a background image or background color, make sure your text is readable.
Meaning List	[a_kind_of(v3,check-110), quantification(v3,plural),object(v3,v118), a_kind_of(v118,be-101), property(v118,clear-55), agent(v118,v109), a_kind_of(v109,text-2), property(v109,your-50),if(v4,v3), a_kind_of(v4,prefer- 104),quantification(v4,plural),for(v4,v35), a_kind_of(v35,apply-107),tense(v35,infinitive), object(v35,v1381),conjunction(v1381,[v60,v92],or),a_k ind_of(v60,image-5), a_kind_of(v60,background-3), quantification(v60,a),a_kind_of(v92,color-5), showing(v75,v92), a_kind_of(v75,background-3), agent(v4,v9),a_kind_of(v9,people-1),anaphoric(v9)]

Table 14. The meaning list for the first sentence of Policy Statement 7: "If you choose to use a background image or background color, make sure your text is readable."

Second Sentence of Policy Statement 7	White or light-colored backgrounds are the most readable.
Meaning List	[a_kind_of(v538,be-101),quantification(v538,plural), property(v538,clear-55),relationship(clear-55,v2), property(v2,'the most'-150),agent(v538,v526), a_kind_of(v526,background-6), quantification(v526,plural),has_property(v526,v3153), conjunction(v3153,[v11,v126],or), property(v11,white-51), property(v126,coloured-51),property(v126,light-55)]

Table 15. The meaning list for the second sentence of Policy Statement 7: "White or light-colored backgrounds are the most readable."

b) Authors Intention. This statement of policy requires that text be readable against whatever background the designer chooses. The author of this policy statement comments that light-colored backgrounds are the most readable, though many people might disagree with this statement, saying that light text on a dark background (i.e., white text on a black background) is preferable. The only real ambiguity in this statement is exactly what "light-colored" means.

c) Entities, Relationships, and Attributes for First Sentence of Policy Statement 7:

Entities: a_kind_of(v3,check-110),
a_kind_of(v118,be-101),
a_kind_of(v109,text-2),

a_kind_of(v4,prefer-104),
a_kind_of(v35,apply-107),
a_kind_of(v60,image-5),
a_kind_of(v60,background-3),
a_kind_of(v92,color-5),
a_kind_of(v75,background-3),
a_kind_of(v9,people-1)

Relationships: a_kind_of, quantification, object, property, agent, if, for, tense, conjunction, showing,
anaphoric

Attributes:
quantification(v3,plural),
property(v118,clear-55),
property(v109,your-50),
quantification(v4,plural),
tense(v35,infinitive),
quantification(v60,a)

Entities, Relationships, and Attributes for Second Sentence of Policy Statement 7:

Entities: a_kind_of(v538,be-101),
a_kind_of(v526,background-6)

Relationships: a_kind_of, quantification, property, relationship, agent, has_property, conjunction

Attributes:
quantification(v538,plural),
property(v538,clear-55),
property(v2,'the most'-150),
quantification(v526,plural),
property(v11,white-51),
property(v126,coloured-51),
property(v126,light-55)

d) Domain of Policy Statement. This statement of policy applies to any regions of a web page that contain text.

e) Modal Auxiliaries. The phrase "make sure that X is Y" is a polite way of saying "X must be Y," and is thus a modal auxiliary.

f) Fuzzy Descriptors. Fuzzy descriptors are "background," "readable," "white," "light-colored," and "most."

g) Anaphoric References. None. The words "you" and "your" refer to the reader or to a hypothetical web-page designer, and are not anaphoric.

h) Speech Act Theory. The phrase "make sure that X is Y" is a polite way of saying "X must be Y." The implication is that white or light-colored backgrounds must be used.

i) Miscellaneous Comments. An ellipsis phenomena is evident in that background images and colors appear on a web page.

8. Policy Statement 8

a) The Statement and its Meaning List. Policy Statement 8 is: "Limit the number of different font styles and colors on a page. A good rule of thumb is to use no more than three different fonts on a page." The first sentence of this policy statement fits into our taxonomy as a statement of policy, while we classify the second sentence as a statement of advice that can be easily converted into policy. The meaning lists for this policy statement are given in text form in Tables 16 and 17.

First Sentence of Policy Statement 8	Limit the number of different font styles and colors on a page.
Meaning List	[a_kind_of(v5,limit-101), quantification(v5,plural),tense(v5,imperative), object(v5,v17),a_kind_of(v17,number-7), quantification(v17,the), has_property(v38,v17),a_kind_of(v38,face-1), property(v38,different-53), type_of(v38,v42),type_of(v38,v62), a_kind_of(v42,style-3),quantification(v42,plural), a_kind_of(v62,color-6),quantification(v62,plural), on(v38,v89),a_kind_of(v89,page-1), quantification(v89,a),distinct(v42,v62)]

Table 16. The meaning list for the first sentence of Policy Statement 8: "Limit the number of different font styles and colors on a page."

Second Sentence of Policy Statement 8	A good rule of thumb is to use no more than three different fonts on a page.
Meaning List	[a_kind_of(v50,be-102), object(v50,v90),a_kind_of(v90,apply-107), quantification(v90,plural),tense(v90,infinitive), object(v90,v106),a_kind_of(v106,face-1), quantification(v106,plural), property(v106,different-51), quantification(v106,'no more than'), quantity(v106,3),on(v106,v123), a_kind_of(v123,page-1), quantification(v123,a),agent(v50,v1), a_kind_of(v1,rule-6), property(v1,good-52),quantification(v1,a)]

Table 17. The meaning list for the second sentence of Policy Statement 8: "A good rule of thumb is to use no more than three different fonts on a page."

b) Authors Intention. The authors intention here is clear: use no more than three different fonts on a given web page.

c) Entities, Relationships, and Attributes for First Sentence of Policy Statement 8

Entities: a_kind_of(v5,limit-101),
a_kind_of(v17,number-7),
a_kind_of(v38,face-1),
a_kind_of(v42,style-3),
a_kind_of(v62,color-6),
a_kind_of(v89,page-1)

Relationships: a_kind_of, quantification, tense, object, has_property, property,type_of, on, distinct

Attributes:
quantification(v5,plural),
tense(v5,imperative),
quantification(v17,the),
property(v38,different-53),
quantification(v42,plural),
quantification(v62,plural),
quantification(v89,a)

Entities, Relationships, and Attributes for Second Sentence of Policy Statement 8

Entities: a_kind_of(v50,be-102),
a_kind_of(v90,apply-107),
a_kind_of(v106,face-1),
a_kind_of(v123,page-1),
a_kind_of(v1,rule-6)

Relationships: a_kind_of, object, quantification, tense, property, quantity, on, agent

Attributes:

quantification(v90,plural),
tense(v90,infinitive),
quantification(v106,plural),
property(v106,different-51),
quantification(v106,'no more than'),
quantity(v106,3),on(v106,v123),
quantification(v123,a),
property(v1,good-52),
quantification(v1,a)

d) Domain of Policy Statement. This policy statement applies to individual web pages.

e) Modal Auxiliaries. None.

f) Fuzzy Descriptors. The word "limit" is a fuzzy descriptor. Note that the term "a good rule of thumb is to use X" should be interpreted as "use X," and so "a good rule of thumb" is not fuzzy.

g) Anaphoric References. The word "use" in the second sentence is an anaphoric reference to the word "limit" in the first sentence.

h) Speech Act Theory. The policy statement can be interpreted as saying "use no more than three different fonts on a page."

i) Miscellaneous Comments. An ellipsis phenomena is evident in that "page" refers to a "web page."

9. Policy Statement 9

a) The Statement and its Meaning List. Policy Statement 10, "Text and graphics that move can be particularly annoying. Use blinking text, scrolling marquees, animated gifs and Java applets very sparingly, if at all." The first sentence of this policy statement fits into our taxonomy as a statement of opinion, and the second sentence fits into our taxonomy as a statement of advice that can be interpreted as policy. The meaning lists for this policy statement are given in text form in Tables 18 and 19. The meaning list for the first sentence of this policy statement is given in graphical format in Figure 15.

First Sentence of Policy Statement 9	Text and graphics that move can be particularly annoying.
Meaning List	[a_kind_of(v50,be-101), tense(v50,can), property(v50,annoying-51), relationship(annoying-51,v92), property(v92,particularly-151), agent(v50,v2), agent(v50,v8), a_kind_of(v2,text-2), a_kind_of(v8,art-1), agent(v10,v2), agent(v10,v8), a_kind_of(v10,move-105), quantification(v10,plural), distinct(v2,v8)]

Table 18. The meaning list for the first sentence of Policy Statement 9: "Text and graphics that move can be particularly annoying."

Second Sentence of Policy Statement 9	Use blinking text, scrolling marquees, animated gifs, and Java applets very sparingly, if at all.
Meaning List	[a_kind_of(v15,apply-107), quantification(v15,plural),object(v15,v31), object(v15,v40),object(v15,v56),object(v15,v62), a_kind_of(v31,text-2),agent(v25,v31), a_kind_of(v25,flash-102),tense(v25,prespart), a_kind_of(v40,marquee-1), quantification(v40,plural),agent(v37,v40), a_kind_of(v37,scroll-100), tense(v37,prespart),a_kind_of(v56,gif-0), quantification(v56,plural),object(v44,v56), a_kind_of(v44,animate-102), tense(v44,pastpart),a_kind_of(v62,applet-0), quantification(v62,plural),has_property(v62,v59), a_kind_of(v59,'Java'-0),property(v15,much-152), conjunction(v15,[v70,v75],if),property(v70,sparingly-150), distinct(v31,v40),distinct(v31,v56), distinct(v31,v62),distinct(v40,v56), distinct(v40,v62),distinct(v56,v62)]

Table 19. The meaning list for the second sentence of Policy Statement 9: "Use blinking text, scrolling marquees, animated gifs, and Java applets very sparingly, if at all."

b) Authors Intention. This statement can reasonably be interpreted as: "Do not use text or graphics that move." The authors intention here is clear.

c) Entities, Relationships, and Attributes for First Sentence of Policy Statement 9.

- Entities:
  - a_kind_of(v50,be-101),
  - a_kind_of(v2,text-2),
  - a_kind_of(v8,art-1),
  - a_kind_of(v10,move-105)
- Relationships:
  - a_kind_of,

    - tense,
    - property,
    - relationship,
    - agent,
    - quantification,
    - distinct
- Attributes:
    - tense(v50,can),
    - property(v50,annoying-51),
    - relationship(annoying-51,v92),
    - property(v92,particularly-151),
    - quantification(v10,plural)

Entities, Relationships, and Attributes for Second Sentence of Policy Statement 9.

- Entities:
    - a_kind_of(v15,apply-107),
    - a_kind_of(v31,text-2),
    - a_kind_of(v25,flash-102),
    - a_kind_of(v40,marquee-1),
    - a_kind_of(v37,scroll-100),
    - a_kind_of(v56,gif-0),
    - a_kind_of(v44,animate-102)
    - a_kind_of(v62,applet-0)
- Relationships: a_kind_of, quantification, object, agent, tense, has_property, property, conjunction, distinct
    - Attributes:
    - quantification(v15,plural),
    - tense(v25,prespart),
    - quantification(v40,plural),
    - tense(v37,prespart),
    - quantification(v56,plural),
    - tense(v44,pastpart),
    - quantification(v62,plural),
    - property(v15,much-152),
    - property(v70,sparingly-150)

d) Domain of Policy Statement: Any page on a web site where text or graphics might appear. Information about the domain of this policy statement is not contained explicitly within this policy statement.

e) Modal Auxiliaries. The word “can” in the first sentence is a modal auxiliary.

f) Fuzzy Descriptors. The terms “particularly,” “annoying,” “very,” “sparingly,” and “if at all’ are fuzzy descriptors.

g) Anaphoric References. "Blinking text, scrolling marquees, animated gifs, and Java applets" are anaphoric references to "text and graphics that move."

h) Speech Act Theory. The first sentence is a speech act requiring that text and graphics that move be avoided or (preferably) not used at all.

i) Miscellaneous Comments. An ellipsis phenomena is evident in that "text and graphics that move," and "blinking text, scrolling marquees, animated gifs, and Java applets" all appear on web pages.

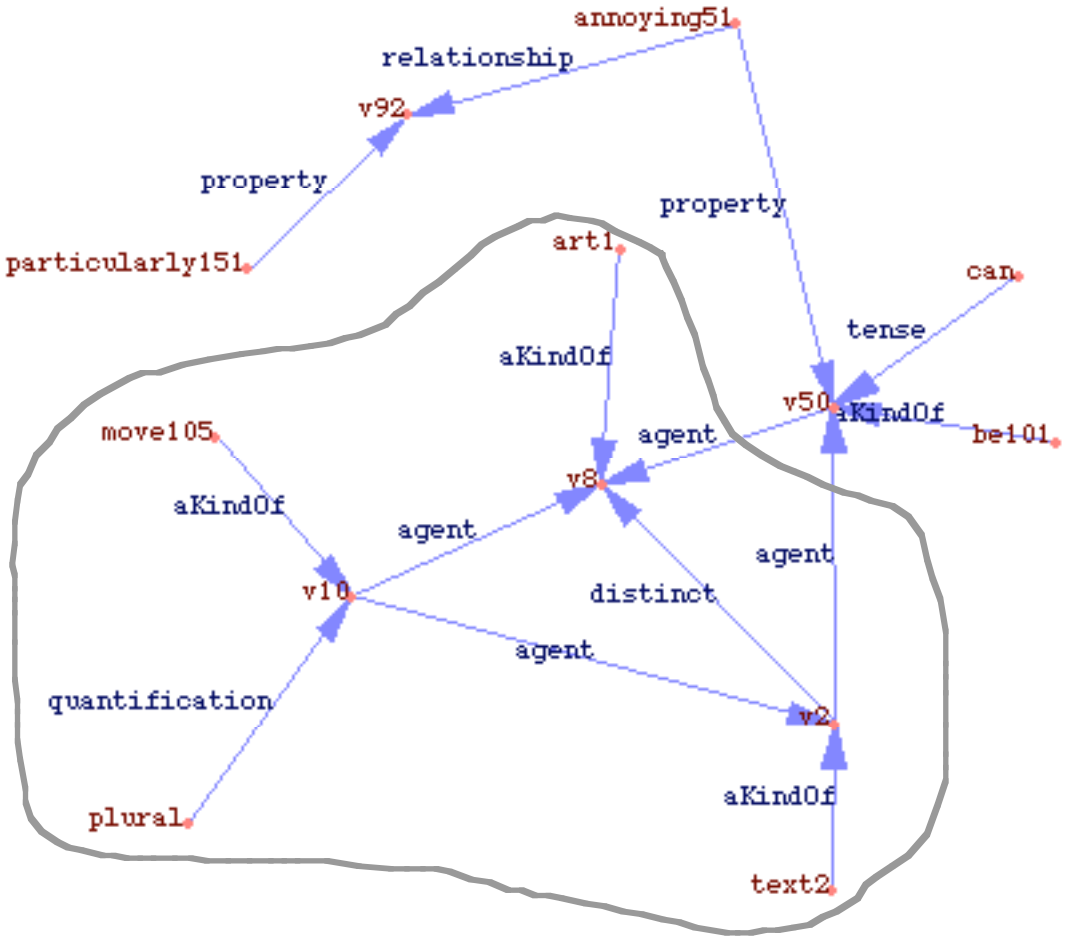

Figure 15. Graphical representation of the meaning list for the first sentence of Policy Statement 9: "Text and graphics that move can be particularly annoying." A semantic network representation for "text and graphics that move" is enclosed in the grey loop.

10. Policy Statement 10

a) The Statement and its Meaning List. Policy Statement 10 is "Any web site collecting personal information must comply with the provisions of reference (d). Network Identification and Internet Protocol addresses are not considered personal data." fits into our taxonomy as an unambiguous statement of policy. The meaning lists for this policy statement are given in text format in Tables 20 and 21. The meaning lists for this policy statement are given in graphical format in Figures 16 and 17.

First Sentence of Policy Statement 10	Network identification and Internet protocol addresses are not considered personal data.
Meaning List	[a_kind_of(v42,see-108), property(v42,not-151), tense(v42,past), object(v42,v79), a_kind_of(v79,data-1), property(v79,personal-55), object(v42,v24), object(v42,v9), a_kind_of(v9,identification-3), agent(v9,v1), a_kind_of(v1,network-2), a_kind_of(v24,address-1), quantification(v24,plural), object(v22,v24), a_kind_of(v22,protocol-1), object(v22,v20), a_kind_of(v20,'Internet'-0), distinct(v9,v24)]

Table 20. Meaning list for the first sentence of Policy Statement 10: "Network identification and Internet protocol addresses are not considered personal data."

Second Sentence of Policy Statement 10	Any web site collecting personal information must comply with the provisions of reference (d).
Meaning List	[a_kind_of(v68, conform102), quantification(v68, plural), tense(v68, must), instrument_of(v68, v92), a_kind_of(v92, provision1), quantification(v92, plural), quantification(v92, the), part_of(v92, v101), a_kind_of(v101, reference3), identification(v101, v116), a_kind_of(v116, d0), agent(v68, v20), a_kind_of(v20, location2) located_at(v10, v20), a_kind_of(v10, web0), quantification(v20, any), agent(v30, v20), a_kind_of(v30, compile101), tense(v30, prespart), object(v30, v58), a_kind_of(v58, information2), property(v58, personal51)]

Table 21. Meaning list for the second sentence of Policy Statement 10: "Any web site collecting personal information must comply with the provisions of reference (d)."

b) Authors Intention. The intention of the author is clear: the cited items are not personal data. The implication is that other statements of policy may put constraints on how personal data should be handled, but Network ID and IP addresses are not subject to these constraints.

c) Entities, Relationships, and Attributes for First Sentence of Policy Statement 10.

- Entities:
  - a_kind_of(v42,see-108),
  - a_kind_of(v79,data-1),
  - a_kind_of(v9,identification-3),
  - a_kind_of(v1,network-2),
  - a_kind_of(v24,address-1),
  - a_kind_of(v22,protocol-1),
  - a_kind_of(v20,'Internet'-0)
- Relationships:
  - a_kind_of,
  - tense,
  - object,
  - property,
  - agent,
  - quantification,
  - distinct
- Attributes:
  - property(v42,not-151),
  - tense(v42,past),
  - property(v79,personal-55),
  - quantification(v24,plural)

Entities, Relationships, and Attributes for Second Sentence of Policy Statement 10.

- Entities:
  - a_kind_of(v68, conform102),
  - a_kind_of(v92, provision1),
  - a_kind_of(v101, reference3),
  - a_kind_of(v116, d0),
  - a_kind_of(v20, location2)
  - a_kind_of(v10, web0),
  - a_kind_of(v30, compile101),
  - a_kind_of(v58, information2)
- Relationships:
  - a_kind_of
  - quantification
  - tense
  - instrument_of
  - part_of, identification
  - agent, located_at
  - object
  - property
- Attributes:
  - quantification(v68, plural),

- tense(v68, must),
- quantification(v92, plural),
- quantification(v92, the),
- tense(v30, prespart),
- property(v58, personal51)

d) Domain of the Policy Statement. The domain of this policy statement consists of any place on a web site where Network Identification or IP addresses might appear– specifically, individual web pages.

e) Modal Auxiliaries. The word "must" in the second sentence is a modal auxiliary.

f) Fuzzy Descriptors. The word "personal" is a fuzzy descriptor.

g) Anaphoric References. The term "personal information" appearing in the second sentence is an anaphoric reference to the term "personal data" of the first sentence.

h) Speech Act Theory. No significant speech acts are apparent here. It's worth noting however that the term "considered" contributes no meaning; the first sentence of this policy statement carries the exact same meaning without this word. Similarly, the phrase "the provisions of" in the second sentence can be eliminated without changing its meaning. In both cases these terms do not contribute any meaning, but may add a small amount of redundancy to aid the reader.

i) Miscellaneous Observations. One way to simplify this statement is to recognize that it consists essentially of two independent statements of policy: first, that network identification is not considered to be personal data, and second, that IP addresses are not to be considered personal data. This is obvious from the natural-language statement itself, and also from the triangular feature in the center of the meaning lists graphical representation. A transformation into two separate meaning lists can be made by (1) eliminating the term distinct(v9,v24) from the meaning list, then (2) grouping together v9, v42, and everything connected to them except for v24, into a single meaning list, and then finally (3) grouping together v24, v42, and everything connected to them

except for v9. As a rule expressed symbolically in terms of natural-language text, we can say

□ and □ are □ □ □ is □, □ is □

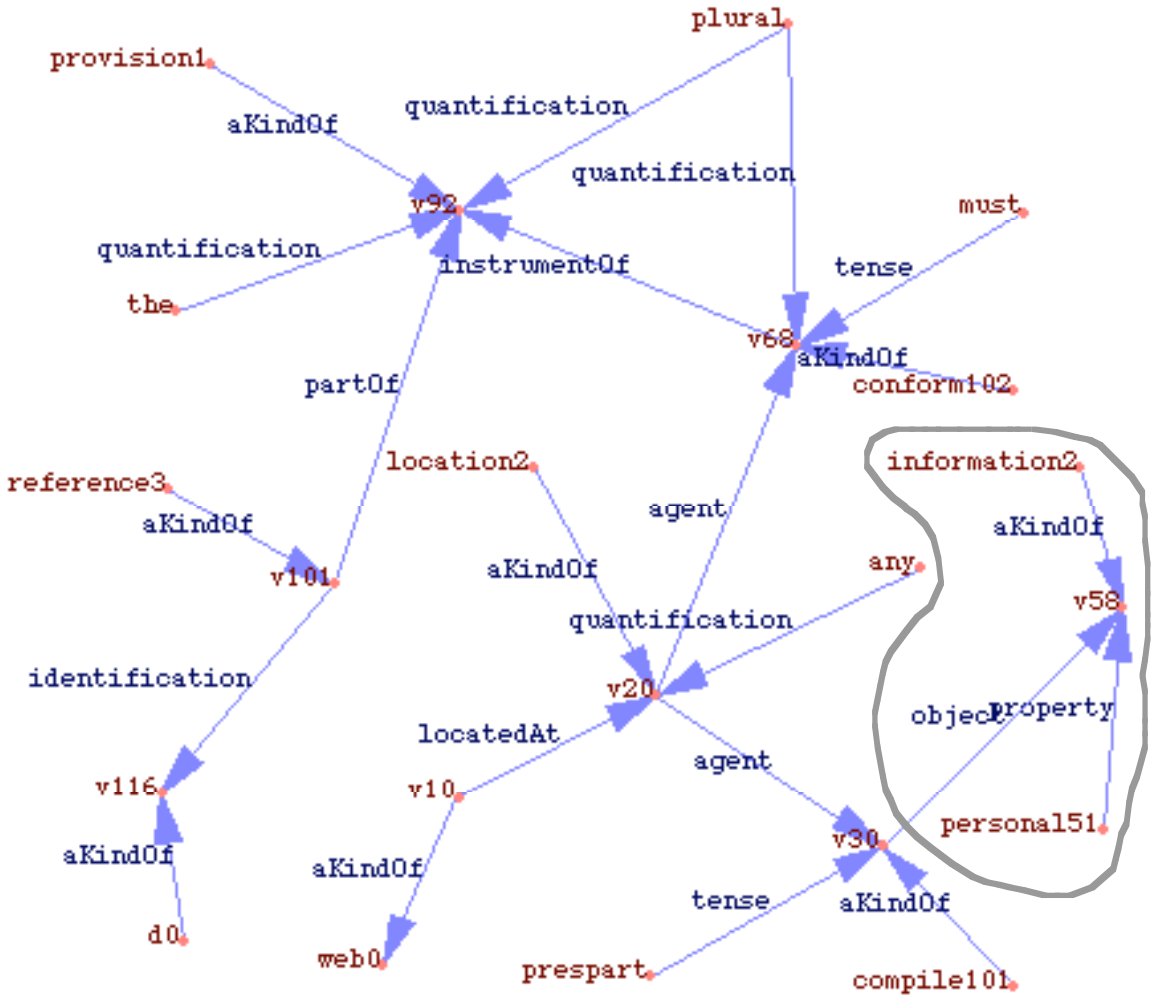

Figure 16. Graphical representation of meaning list for Policy Statement 10.1: "Any web site collecting personal information must comply with the provisions of reference (d)." A semantic network representation for "personal information" is enclosed in the grey loop.

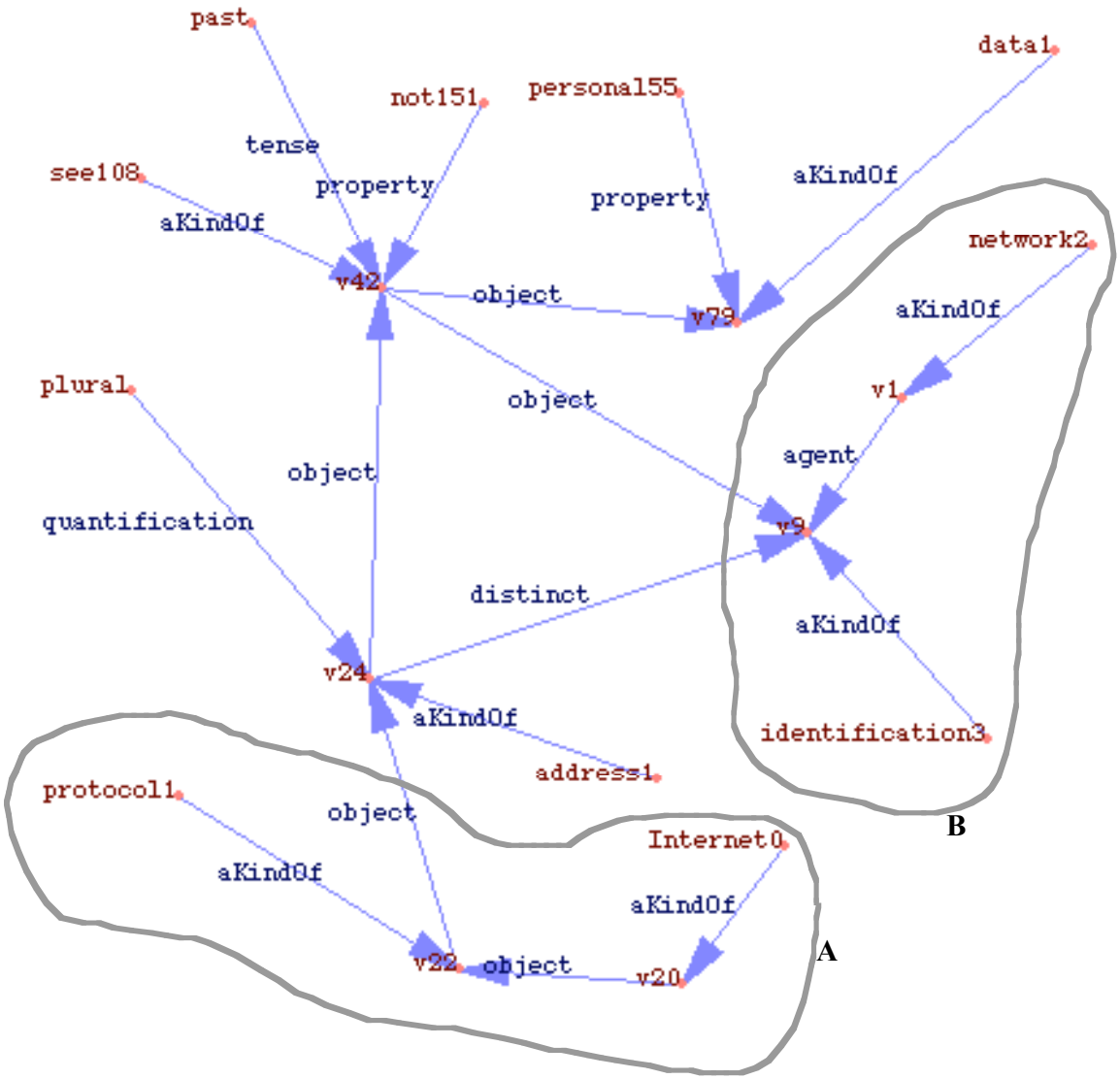

Figure 17. Graphical representation of the meaning list for "Network Identification and Internet Protocol addresses are not considered personal data." A semantic network representation for a "Network Identification" object is enclosed in the grey loop labeller "A," and a semantic network representation for an "Internet Protocol address" object is enclosed in the grey loop labeller "B."

THIS PAGE IS INTENTIONALLY LEFT BLANK

V. DISCUSSION OF RESULTS

Our case study was carried out to characterize the ambiguities that appear in natural-language policy statements, and the feasibility of resolving these ambiguities automatically in a fielded system. Our case study focused on ambiguities of the following types.

- Anaphoric References. It was observed that anaphoric references were relatively simple, in part because it was unusual for policy statements to extend over more than two sentences. It should be straightforward to resolve anaphoric references in natural-language policy statements using standard algorithms such as those discussed in [Jurafsky and Martin, 2000], [Lappin and Leass, 1994], and [Hobbs, 1978].
- Speech Acts. For the most part, speech acts within our case study consisted of suggestions or statements of advice which were intended to be interpreted as imperative commands. It is not expected that speech acts will present significant difficulties for a natural language input system for a policy workbench.
- Fuzzy Expressions. We identified over two dozen fuzzy expressions in the ten natural-language statements of policy that we studied. As shown in Table 3, these fuzzy expressions covered a variety of grammatical categories, and fell along a number of different dimensions. Despite the variety of fuzzy expressions found in our case study, the discussion following Table 3 demonstrates that each can be quantified in some reasonable way.

Our case study also identified modal auxiliaries contained within policy statements. Modal auxiliairies are significant because they provide important clues for the semantic interpretation of policy statements.

Determining the domain to which a policy statement applies is important for analysis of policy statements, but reliably identifying domain objects through automatic means appears to be a non-trivial problem. Some statements of policy do not even mention the domain explicitly, so that the reader is required to infer the object (or objects) to which it applies. For example, Policy Statement 6 (“Use numbered or bulleted

lists to condense text and to break up the page visually") applies to web pages, though "web pages" are not mentioned in the statement. Interestingly, however, there were only a small number of domain objects for the statements from our case study: web pages, and web sites. We expect that this will be the case in general, so that the relevant domain only needs to be chosen from a limited number of possibilities.

VI. CONCLUSIONS

In this paper we introduced and discussed a novel architecture for the analysis of natural language policy statements. The architecture employs semantic networks to represent natural language policy statements, and uses a "graph-matching" technique to determine whether a particular domain object conforms to or violates a given statement of policy. The benefits associated with our architecture include representations that lend themselves well to visualization in a graphical form, and a proof technique that promises low computational complexity in practical applications.

After discussing our architecture, we carried out a case study which focused on identifying and categorizing the various ambiguities appearing in natural-language policy statements. The case study analyzed ten natural-language policy statements culled from a policy document for web sites hosted by a university. Our findings suggest that algorithms will be able to resolve ambiguities due to anaphora, speech acts, and fuzzy expressions in most natural language policy statements.

There are several important directions in which our work can be extended. The work contained in this paper has shown the feasibility of a novel architecture for analysis of natural-language policy statements, but among the work that needs to be done before construction can be considered is a rigorous comparison between our logic based on semantic networks and the first-order predicate calculus. It should be possible to compare the representations and proof techniques of these two formalisms through a careful reading of the relevant literature; that is, it is not expected that any original research would need to be done. However, it is important that this literature be reviewed and the findings documented to establish a strong theoretical foundation for continued work.

Another area where work remains to be done is with our use of Mathematica for generating graphical representations of meaning lists. We had hoped that the code which embeds graphs in the 2-dimensional plane would be able to generate clear graphical representations of meaning lists in a fully automated manner. Unfortunately, the graphical representations that are generated automatically usually have minor but annoying imperfections: labels that obscure each other, edges that needlessly cross, or

other problems that require editing by a human. Though the editing process is relatively quick and painless, and an argument could be made that working with a graph in the editor is an effective way of gaining important information about it, it would still be better if this work were done automatically.

Also, our graphical representations were generated by writing the outputs of the MARIE parser to a file, performing some simple edits on the file using a search-and-replace utility, and then reading the file into Mathematica. A version of our software that ran without the Mathematica interpreter and that could be integrated directly into the MARIE parser would be very helpful for further work in this area.

APPENDIX A: CONVERSION OF MEANING LISTS INTO GRAPHICAL FORM

This appendix documents a Mathematica package that converts meaning lists from the MARIE parser into a graphical form. The package, called `nlpPlotting`, is shown in Figure 18 below.

To use the package, it is read into Mathematica, and the command `Get[name]`, often seen in the short form `<<name`, is used to read in the text of the file containing the meaning list information. To generate this file, the output from the MARIE parser has to be edited slightly: the expression `result()` wrapped around meaning lists must be eliminated; square brackets `[]` enclosing meaning list components must be replaced by curly brackets `{}` to create Mathematica lists; parentheses `()` in meaning-list attributes, entities, and relationships must be replaced by square brackets `[]` to create Mathematica expressions; and all dashes and underscores in meaning lists must be dropped to prevent conflict with the special meanings that Mathematica has for these symbols.

Once the meaning lists are read in, they are converted into graph data-structures by the function `generateMLGraph[]`. They can subsequently be plotted using the `ShowGraph[]`, the `SpringEmbedding[]`, `RadialEmbedding[]`, `RankedEmbedding[]`, `RootedEmbedding[]`, and `ShakeGraph[]` commands, as explained in [Skiena, 1990]. A simple Java-based editor which allows the nodes of a graph to be moved without "breaking" edges can be obtained from

http://www.cs.sunysb.edu/~lloyd/grapheditor/index.html

Our package consists essentially of one function, `generateMLGraph[]`, and two helper functions, `binaryRelationQ[]`, and `meaningListQ[]`. The function `generateGraph[]` converts a meaning list into a graph data structure that can be manipulated by commands in the Combinatorica package. The helper functions ensure that the code given below for `generateGraph[]` is called only when its input has the correct form for a meaning list.

Before defining any functions, however, we set up the "framework" required for proper use of a package. This framework, consisting of lines 1, 5, 23, and 24, set up both a public and a private context that our functions will use. A context is essentially a directory structure for variable names, which helps to prevent collisions of variable names [Wolfram, 1996]. The usage messages in lines 2, 3, and 4 appear in the public context, and provide a user with a short summary of a function.

```
(*
The function generateGraph converts a meaning list in text format into
a graph that can be plotted with the Combinatorica graph-plotting
commands.  The meaning list is assumed to come from the MARIE parser
after all underscore characters "_" have been removed from variable
names.

The function meaningListQ determines whether an input argument has the
form of a meaning list.

The function binaryRelationQ determines whether an input argument has
the form of a binary relation. *)

(*1*)  BeginPackage["mlPlotting`"]

(*2*)  generateMLGraph::usage="generateMLGraph[s_List] converts \
a meaning list into a graph that can be plotted using Combinatorica's \
ShowGraph command."

(*3*)  binaryRelationQ::usage="binaryRelationQ[e_] returns true if the\
expression e is a binary relation; that is, if it is not unary, \
tertiary, etc.  It returns false otherwise."

(*4*)  meaningListQ::usage="meaningListQ[e_] returns true if the \
expression e has the form of a meaning list; that is, if e is a \
list of binary relations.  It returns false otherwise."

(*5*)  Begin["`Private`"]

(*6*)  Needs["DiscreteMath`Combinatorica`"];

(*7*)  binaryRelationQ[e_]:= Length[e]==2;

(*8*)  meaningListQ[e_]:= And@@(binaryRelationQ/@e)&&(Head[e]===List);

(*9*)  generateMLGraph[s_/;meaningListQ[s]]:=
(*10*)       Module[
(*11*)       {meaningList,relationships,entities,edgeDataStructPrelim,
(*12*)       edgeDataStruct,edgeIndexRules,g},
(*13*)       meaningList=s;
(*14*)       relationships=Union[Head/@meaningList];
(*15*)       entities=Union[Flatten[{#[[1]],#[[2]]}&/@meaningList]];
(*16*)       g=EmptyGraph[Length[entities]];
```

```
(*17*)       AppendTo[g[[2,#]],VertexLabel->
                  entities[[#]]]&/@Range[Length[entities]];
(*18*)       edgeDataStructPrelim=meaningList/.
                  a_[b_,c_]->{{b,c},EdgeLabel->a};
(*19*)       edgeIndexRules=Transpose[
                  {entities,Range[Length[entities]]}]/.{a_,b_}->(a->b);
(*20*)       edgeDataStruct=edgeDataStructPrelim/.edgeIndexRules;
(*21*)       Return[AddEdges[g,edgeDataStruct]];
(*22*) ];

(*23*) End[]  (*end private context*)

(*24*) EndPackage[]
```

Figure 18. The mlPlotting package. This package is contained in a plain text file called `mlPlotting.m`, which is meant to suggest the phrase "Meaning List Plotting."

Within the private context (line 5), we read in the Combinatorica package so that we have available functions like `EmptyGraph[]` and `AddEdges[]`. The function `generateMLGraph[]` converts a meaning list into a Graph data structure that can be manipulated by Combinatorica. We want this function to accept as an argument only expressions that have the form of a meaning list, and we want arguments of any other form to leave the function unevaluated. We specify that a "meaning list" has the form of a list of binary expressions of arbitrary length (there may possibly be exceptions to this, but they are rare and we leave this topic for future research). The function `meaningListQ[]` in line 8 returns True or False, according to whether or not its argument is a "meaning list quantity." It uses the function `binaryRelationQ[]` to determine whether an expression is a "binary relation quantity."

The function `generateMLGraph[]` is defined in lines 9-22. It accepts as an argument any expression (which we call `s`), subject to the condition (indicated by `/;`) that `meaningListQ[s]` returns `True`. The function code is enclosed within a `Module[]`, which treats the variables in the initial list (lines 11 and 12) as local. A local copy of the input meaning list is made in line 13, and all the relationships in the meaning list are extracted in line 14. To clarify the terminology being used here, we consider a meaning list to be a collection of entities, with relationships specified between them. The variable names in the package were chosen to reflect their association with the entities

and relationships within a meaning list, where the entities and relationships within a meaning list are as shown in the following expression.

```
{relationship1[entity1a,entity1b],relationship2[entity2a,en
tity2b],…,relationshipN[entityNa,entityNb]}
```

Thus, getting a list of all the relationships is a simple matter of getting the "heads" of all the binary relationships. This is done in line 14 by mapping the function `Head[]` onto all the expressions in the meaning list; this is accomplished with the `Map` operator (which has `/@` as a short form). Similarly, getting a list of all the entities in line 15 involves mapping an appropriate function over the `meaningList` variable; in this case, we use an anonymous function, expressed by `{#[[1]],#[[2]]}&`, that simply returns a list containing the two arguments of the expression it is applied to. The functions `Union[]`, and `Flatten[]` are used to remove redundant terms and unwanted levels of nesting.

In line 16 we generate an empty graph `g` with the same number of vertices as the number of unrepeated entities we have, and in line 17 we assign the appropriate label to each vertex in the graph `g`. Lines 18, 19 and 20 work together to generate a data structure for the edges of the graph. We begin in line 18 by generating a modified form of the meaning list: each binary relation is converted into a list that defines two vertices connected by an edge, and the label that goes on that edge. Line 19 is the definition of a set of rules that transform entity names into a sequence of numbers; this is necessary because each vertex is fundamentally represented by a number, with the entity name being nothing more than a label. Line 20 applies these rules to generate the final data structure representation for edges. Line 21 adds these edges to the graph `g`, resulting in the finished graph data structure.

LIST OF REFERENCES

Allen, James, *Natural Language Understanding*, The Benjamin/Cummings Publishing Company, 1995.

Brennan, S. E., M. W. Friedman, and C. Pollard, "A centering approach to pronouns," ACL-87, Stanford, California, 1987, pp. 155-162.

Berzins, Valdis, and Luqi, *Software Engineering with Abstractions*, Addison Wesley Publishing Company, 1990.

Bloomfield, Leonard, *Language*, The University of Chicago Press, Chicago, 1984.

Brooks, Frederick P. Jr., *The Mythical Man-Month*, Addison Wesley Longman, Inc., 1995.

Covington, Michael A., *Natural Language Processing for Prolog Programmers*, Prentice Hall, Englewood Cliffs, New Jersey, 1994.

Damianou, Nicodemos C., *A Policy Framework for Management of Distributed Systems*, Dissertation, University of London, Imperial College of Science, Technology and Medicine, 2002.

Damianou, Nicodemos, Naranker Dulay, Emil Lupu, and Morris Sloman, *The Ponder Policy Specification Language*, In Sloman, M., Lobo, J. and Lupu, E. C., eds., Lecture Notes in Computer Science, No. 1995: Proceedings of the International Workshop on Policies for Distributed Systems and Networks, Springer-Verlag, January 2001, pp. 18-38.

Guglielmo, E. J., and N. C. Rowe, "Natural language retrieval of images based on descriptive captions." ACM Transactions on Information Systems, 14, 3 (May 1996), pp. 237-267.

Hobbs, J. R., "Resolving Pronoun References." *Lingua, 44*, 1978, pp. 311-338.

Hodges, John C. and Mary E. Whitten, *Hodges' Harbrace College Handbook*, Harcourt Brace Jovanovich, Inc., New York, 1982.

Jurafsky, Daniel, and James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, Upper Saddle River, New Jersey, 2000.

Johnsonbaugh, Richard, *Discrete Mathematics*, Macmillan Publishing Company, New York, 1984.

Joyce, Steven T., and John Q. Walker, "Policy-Based Network-Management: Getting Started." White paper, NetIQ Corporation, July 1999. A version of this paper appeared in *Cisco World*, October 1999, pp. 18-20.

Lappin, S. and H. Leass, "An algorithm for pronominal anaphora resolution." Computational Linguistics, 20(4), 1994, pp. 535-561.

McCawley, James D., *The Syntactic Phenomena of English, Second Edition*, The University of Chicago Press, Chicago, 1998.

McCawley, James D., *Everything That Linguists Have Always Wanted to Know About Logic But Were Ashamed to Ask, Second Edition*, The University of Chicago Press, Chicago, 1993.

Meyer, J. J. Ch., R. J. Wieringa, and F. P. M. Dignum, "The Role of Deontic Logic in the Specification of Information Systems." Utrecht University, Department of Computer Science Document Number UU-CS-1996-55, ISSN: 0924-3275, December 1996.

Michael, James B., Edgar H. Sibley, Richard F. Baum, Richard L. Wexelblat, and Fu Li, "Experiments in Support of Policy Representation." Proceedings of the International Conference on Economics/Management and Information Technology. Japan Society for Management Information, Tokyo, 1992, pp. 323-326.

Michael, James Bret, Vanessa L. Ong, and Neil C. Rowe, "Natural Language Processing Support for Developing Policy-Governed Software Systems." 39th International Conference on Object-Oriented Languages and Systems, Santa Barbara, California, July-August 2001.

Miller, George A., "Nouns in WordNet: A Lexical Inheritance System." Five Papers on WordNet, pp. 10-25, 1993. Available at http://www.cogsci.princeton.edu/~wn/5papers.pdf.

New York Times, *Confusion Over Policy is Major Issue in Train Death*, August 1, 2002.

Ong, Vanessa L. "An Architecture and Prototype System for Automatically Processing Natural-Language Statements of Policy." Thesis, Naval Postgraduate School, Monterey, California, March 2001.

Pinker, Steven, *The Language Instinct: How the Mind Creates Language*, HarperPerennial, New York, 1995.

Rowe, N. C., "Precise and efficient retrieval of captioned images: The MARIE project." Library Trends, 48, 2 (Fall, 1999), pp. 475-495.

Russell, Stuart, and Peter Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

Sibley, Edgar H., James Bret Michael, and Richard L. Wexelblat, "Use of an Experimental Policy Workbench: Description and Preliminary Results." Database Security, V: Status and Prospects, C. E. Landwehr and S. Jajodia (Eds.), Elsevier Science Publishers, Amsterdam, The Netherlands, 1992, pp. 47-76.

Skiena, Steven, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Addison-Wesley Publishing Company, Redwood City, California, 1990.

Sloman, M., J. Lobo, and E. C. Lupu, eds., Lecture Notes in Computer Science, No. 1995: Proceedings of the International Workshop on Policies for Distributed Systems and Networks, Springer-Verlag, January 2001.

Wolfram, Stephen, *The Mathematica Book, 3rd ed.*, Wolfram Media/Cambridge University Press, 1996.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Bret Michael
   Naval Postgraduate School
   Monterey, California

4. Professor Neil C. Rowe
   Naval Postgraduate School
   Monterey, California